Annals of Pure and Applied Logic 54 (1991) 17–58
North-Holland

Polynomial-time versus recursive models

Douglas Cenzer
University of Florida, Gainesville, FL 32611, USA

Jeffrey Remmel*
University of California at San Diego, La Jolla, CA 92093, USA

Communicated by A. Nerode
Received 1 September 1989
Revised 23 January 1990

Abstract

Cenzer, D. and J. Remmel, Polynomial-time versus recursive models, Annals of Pure and Applied Logic 54 (1991) 17–58.

The central problem considered in this paper is whether a given recursive structure is recursively isomorphic to a polynomial-time (p-time) structure. Positive results are obtained for all relational structures, for all Boolean algebras and for the natural numbers with addition, multiplication and the unary function 2^x. Counterexamples are constructed for recursive structures with one unary function and for Abelian groups and also for relational structures when the universe of the structure is fixed. Results are also given which distinguish primitive recursive structures, exponential-time structures and structures with honest witnesses.

Introduction

The basic question which motivated this paper is the following. What are the differences between recursive structures and polynomial time structures? A recursive structure $\mathscr{A}$ over a language $\mathscr{L}$ consists of a recursive subset A of the natural numbers $\mathbb{N}$ which is called the universe of $\mathscr{A}$ together with interpretations of the constant symbols, relation symbols, and function symbols of $\mathscr{L}$ such that each relation symbol is interpreted by a recursive relation and each function symbol is interpreted by a partial recursive function. Note that if $\mathscr{A}$ is an infinite structure, then $\mathscr{A}$ is recursively isomorphic to a recursive structure $\mathscr{B}$ whose universe is $\mathbb{N}$, since each infinite recursive set is the image of $\mathbb{N}$ under a 1-to-1 recursive function. Thus when considering infinite recursive structures, one normally identifies a given recursive structure with a recursively isomorphic copy

* This research was partially supported by NSF grant DMS-87-02473.

whose universe is just the set of natural numbers. Mimicking the definition of recursive structure, we say that a polynomial-time structure $\mathscr{A}$ consists of a polynomial-time subset A of $\{0,1\}^*$ called the universe of $\mathscr{A}$ together with interpretations for the constant symbols, relation symbols, and function symbols such that each relation symbol is interpreted by a polynomial-time relation and each function symbol is interpreted by a function which is the restriction of a polynomial-time function to the appropriate domain. It is not the case that any two infinite polynomial-time sets are polynomial-time isomorphic. Hence it no longer seems reasonable that, when we deal with polynomial time structures we can always identify a given polynomial-time structure with a polynomial-time structure whose universe is $\{0,1\}^*$. (In fact, it makes a difference whether a natural number is given a unary (or tally) representation as a string of 1's, or is given a binary representation as a string of 0's and 1's. This is why we use $\{0,1\}^*$ as the general universe rather than $\mathbb{N}$.) Thus one of the first questions that comes to mind is: When is a recursive structure isomorphic, or recursively isomorphic, to a polynomial time structure?

Here are two examples which illustrate both the negative and positive outcomes to this question.

Consider first the following simple example. Let $\mathscr{A} = (A, 0, S, R)$, where $A = \{0\}^*$ (that is, the set of natural numbers in unary representation), S is the successor function (that is, $S(0^n) = 0^{n+1}$), and R is a unary relation (that is, a subset of $\{0\}^*$). Now if $\mathscr{A}$ is isomorphic to a polynomial-time structure $\mathscr{B} = (B, 0^B, S^B, R^B)$, then we can test for membership in R as follows. Given 0^n, compute $(S^B)^n(0^B) = y_n$ and then test whether y_n is in R^B. Now if we assume that we can compute $S^B(x)$ in $|x|^k$ steps for $|x| \geq 2$, then it takes at most $\sum_{i=1}^{n} |0^B|^{k^i} \leq |0^B|^{k^{n+1}}$ steps to compute y_n. Next we may assume that testing whether $x \in R^B$ takes $|x|^r$ steps if $|x| \geq 2$, then we see that it takes at most $|0^B|^{r(k^n+1)}$ steps to test whether 0^n is in R. This means that R is a doubly-exponential-time set. Thus if we start with any recursive structure $\mathscr{A} = (A, 0, S, R)$, where R is a recursive set, but is not doubly exponential-time, then $\mathscr{A}$ is not even isomorphic, much less recursively isomorphic, to a polynomial-time structure.

Despite this example, there are lots of recursive structures which are recursively isomorphic to polynomial-time structures.

Consider for example structures with universe A as above and one unary function f. We say that 0^m and 0^n are in the same f-orbit if, for some $k \geq 0$, either $f^k(0^m) = 0^n$ or $f^k(0^n) = 0^m$. If f is length-increasing, then it is clear that each f-orbit is isomorphic to (A, S). Now let f and g be any two recursive length-increasing functions from $\{0\}^*$ into $\{0\}^*$. Then the structures (A, f) and (A, g) are recursively isomorphic if and only if they have the same number of orbits. Thus, for example, we can let $f(0^n) = 0^{a(n)}$, where a is Ackermann's function, and still be guaranteed that (A, f) is recursively isomorphic to a polynomial-time structure.

Of course, similar questions naturally arise for models in any two different

complexity classes. For example, in this paper we shall consider contrasts between recursive and primitive recursive structures, between primitive recursive and exponential-time structures and between exponential-time and polynomial-time structures. The techniques of our proofs can be applied to a wide variety of complexity-theoretic classes, but we choose to concentrate on the classes above so as not to obfuscate the proofs with excessive machinery.

Our results in this paper fall into two basic categories.

First, we have positive results, which state that certain recursive structures are recursively isomorphic to polynomial-time structures. Here we prove, for example, that all recursive relational structures, all recursive Boolean algebras, and the structure $(\mathbb{N}, S, +, \cdot, 2^x, <)$ are recursively isomorphic to polynomial-time structures.

Next we have negative results. For example, we show that for the language with one unary function symbol, there is a recursive structure which is not recursively isomorphic to any primitive recursive structure and there is an exponential-time structure which is not recursively isomorphic to any polynomial-time structure. We show that if the language has two unary function symbols, or one binary function symbol, then the above results hold where we replace 'recursively isomorphic' with 'Δ_2^0-isomorphic'. We also can construct recursive Abelian groups which are not recursively isomorphic to any primitive recursive Abelian groups and exponential-time Abelian groups which are not recursively isomorphic to any polynomial-time Abelian groups. Finally, we should mention that our positive result that every recursive relational structure is recursively isomorphic to a polynomial-time structure depends crucially on the fact that we allow the universe of a polynomial-time structure to be any polynomial-time subset of $\{0,1\}^*$. In contrast, we show that if you limit a polynomial-time structure to always have universe $\{0,1\}^*$, then the result fails. That is, in particular, there is a recursive linear ordering of type $\omega + \omega^*$ which is not recursively isomorphic to any polynomial-time structure whose universe is $\{0,1\}^*$.

The outline of this paper is as follows. We begin with a section on preliminaries and notation.

Section 2 contains the results on relational structures and Boolean algebras.

Theorem 2.1. *Any recursive structure $\mathcal{M}$ with no functions is recursively isomorphic to a p-time structure (in fact, to a linear-time structure).*

On the other hand, we have

Theorem 2.2. *Let A be any fixed polynomial time subset of $W = \{0,1\}^*$. Then there exists a recursive linear ordering $\mathcal{L} = \langle W, <_W \rangle$ of order type $\omega + \omega^*$ which is not recursively isomorphic to any polynomial time linear ordering $\mathcal{L}'$ with domain A.*

From Theorem 2.1, we obtain the following result for the language $\{\wedge, \vee, \neg\}$ of Boolean algebras.

Theorem 2.6. *Every recursive Boolean algebra $\mathcal{B}$ is recursively isomorphic to a p-time Boolean algebra.*

The notions of honest witnesses and honest polynomial-time structures are defined. It is shown that every recursive Boolean algebra with honest witnesses has a recursive set of atoms and a theorem of Goncharov [2] is applied to obtain:

Theorem 2.7. *There exists a p-time Boolean algebra which is not isomorphic to any p-time Boolean algebra with honest witnesses.*

It is shown that every honest p-time Boolean algebra is finite; on the other hand we have

Theorem 2.8. *There exists an infinite polynomial-time Boolean algebra with honest witnesses.*

Section 3 contains the results on structures with functions and on Abelian groups.

There are several results in the other direction when function symbols are allowed.

Theorem 3.1. (i) *There is a recursive structure with one unary function which is not recursively isomorphic to any primitive recursive structure.*

(ii) *There is an exponential-time structure with one unary function which is not recursively isomorphic to any polynomial-time structure.*

For structures with binary functions or with more than one unary function, we can strengthen Theorem 3.1 as follows.

Theorem 3.3. (i) *There is a recursive structure with two unary functions which is not Δ_2^0-isomorphic to any primitive recursive structure.*

(ii) *There is an exponential-time structure with two unary functions which is not Δ_2^0-isomorphic to any polynomial-time structure.*

Theorem 3.4. (i) *There is a recursive structure with one binary function which is not Δ_2^0-isomorphic to any primitive recursive structure.*

(ii) *There is an exponential-time structure with one binary function which is not Δ_2^0-isomorphic to any polynomial-time structure.*

An Abelian group may be viewed as a structure with one binary function symbol. Then we have

Theorem 3.6. (i) *There is a recursive Abelian group G which is not recursively isomorphic to any primitive recursive Abelian group.*

(ii) *There is an exponential-time Abelian group which is not recursively isomorphic to any polynomial-time Abelian group.*

Section 4 contains the results on the arithmetic of natural numbers. It is shown that certain restricted versions of the standard model of arithmetic are isomorphic to polynomial time structures.

Theorem 4.1. *The structure* $(\mathbb{N}, 0, 1, S, <, +, \cdot, 2^x)$ *is recursively isomorphic to a polynomial-time structure.*

1. Preliminaries

Some definitions are needed. Let Σ be a finite alphabet. Then Σ^* denotes the set of finite strings of letters from Σ and Σ^ω denotes the set of infinite strings, where $\omega = \{0, 1, 2, \ldots\}$. In particular the set ω of natural numbers in unary form will be identified with $\{1\}^*$ and in binary form will be identified with a subset of $\{0, 1\}^*$.

For a string $\sigma = (\sigma(0), \sigma(1), \ldots, \sigma(n-1))$, $|\sigma|$ denotes the length n of σ. The empty string has length 0 and will be denoted by $\emptyset$. A constant string σ of length n will be denoted k^n. For $m < |\sigma|$, $\sigma \mid m$ is the string $(\sigma(0), \ldots, \sigma(m-1))$; σ is an *initial segment* of τ (written $\sigma < \tau$) if $\sigma = \tau \mid m$ for some m. The *concatenation* $\sigma * \tau$ (or sometimes just $\sigma\tau$) is defined by

$$\sigma * \tau = (\sigma(0), \sigma(1), \ldots, \sigma(m-1), \tau(0), \tau(1), \ldots, \tau(n-1)),$$

where $|\sigma| = m$ and $|\tau| = n$; in particular we write $\sigma^\frown a$ for $\sigma * (a)$.

For any finite alphabet Σ, there is a natural embedding of Σ^* into $\{0, 1\}^*$, given as follows. We may suppose that $\Sigma = \{0, 1, 2, \ldots, n\}$ for some n. For $\sigma = (i_1, \ldots, i_k)$, let

$$\rho(\sigma) = 1^{i_1\frown}0 * 1^{i_2\frown}0 * \cdots * 1^{i_k\frown}0.$$

The function ρ is actually an isomorphism from ω^* onto $\{0, 1\}^*$ and has an inverse ρ^{-1}.

The coding function $\langle \sigma_1, \sigma_2, \ldots, \sigma_k \rangle_k$ for $\sigma_1, \ldots, \sigma_k \in \{0, 1\}^*$ is now defined by

$$\langle \sigma_1, \sigma_2, \ldots, \sigma_k \rangle_k = \rho(\sigma_1^\frown 2 * \sigma_2^\frown 2 * \cdots * \sigma_k).$$

Let $Q_k = \{\langle \sigma_1, \sigma_2, \ldots, \sigma_k\rangle_k : \sigma_i \in \{0,1\}^* \text{ for each } i\}$. The projection functions π_i^k from Q_k onto $\{0,1\}^*$ are implicitly defined by the equation

$$\sigma = \langle \pi_1^k(\sigma), \pi_2^k(\sigma), \ldots, \pi_k^k(\sigma)\rangle_k.$$

The superscript k will normally be omitted.

In the special case that n is a natural number in binary form, we decode n into a sequence $((n)_1, \ldots, (n)_k)$ of natural numbers, where $n = \langle (n)_1, \ldots, (n)_k\rangle$ if $n \in Q_k$ and each $(n)_i = \pi_i(n)$ is also a natural number; otherwise each $(n)_i = 0$.

It is easy to see that the functions π_i, ρ, ρ^{-1}, and $\langle \cdot, \cdot \rangle$ and also the sets Q_k are all computable in linear time.

We shall consider structures over an effective language

$$\mathscr{L} = \langle \{R_i\}_{i\in S}, \{f_i\}_{i\in T}, \{c_i\}_{i\in U}\rangle,$$

where S, T and U are initial segments of ω, for all $i \in U$, c_i is a constant symbol and there are partial recursive functions s and t such that, for all $i \in S$, R_i is an $s(i)$-ary relation symbol and, for all $i \in T$, f_i is a $t(i)$-ary function symbol.

Our basic computation model is the standard multitape Turing machine of Hopcroft and Ullman [4]. Note that there are different heads on each tape and that the heads are allowed to move independently. This implies that a strong σ can be copied in linear time.

Let $t(n)$ be a function on natural numbers. A Turing machine M is said to be $t(n)$ time bounded if each computation of M on inputs of size n has length at most $t(n)$. If t can be chosen to be a polynomial, then M is said to be polynomial-time bounded. A function $f(x)$ on strings is said to be polynomial-time (p-time) if there is a polynomial-time bounded Turing machine M which computes $f(x)$ on input x. A set of strings or a relation on strings is p-time if the characteristic function is p-time. Similar definitions can be given for other complexity classes, such as exponential-time.

Let ϕ_i be the partial recursive function computed by the ith Turing machine M_i. Given a string $\sigma \in \{0,1\}^*$, we write $\phi_i^s(\sigma)\downarrow$ if M_i gives an output in s or fewer steps when started on input string σ. Thus the function ϕ_i^s is uniformly polynomial-time. We write $\phi_e(\sigma)\downarrow$ if $(\exists s)(\phi_e^s(\sigma)\downarrow)$ and $\phi_e(\sigma)\uparrow$ if not $\phi_e(\sigma)\downarrow$.

Let Γ be some complexity class of sets (and functions), such as partial recursive, primitive recursive, exponential time, polynomial time (or p-time). We say that a set or function is Γ-computable if it is in Γ.

Definition 1.1. A structure

$$\mathscr{A} = (A, \{R_i^{\mathscr{A}}\}_{i\in S}, \{f_i^{\mathscr{A}}\}_{i\in T}, \{c_i^{\mathscr{A}}\}_{i\in U}),$$

(where the universe A of $\mathscr{A}$ is a subset of Σ^*) is a *Γ-structure* if

(i) A is a Γ-computable subset of Σ^*;

(ii) for each $i \in S$, $R_i^{\mathscr{A}}$ is a Γ-computable relation;

(iii) for each $j \in T$, $f_j^{\mathscr{A}}$ is a Γ-computable function from $A^{t(i)}$ into A. (More formally, the function $\bar{f}_j^{\mathscr{A}}$ from $(\Sigma^*)^{t(i)}$ into Σ^* is Γ-computable, where

$$\bar{f}_j^{\mathscr{A}}(x_1, \ldots, x_{t(j)}) = \begin{cases} f_j^{\mathscr{A}}(x_1, \ldots, x_{t(j)}), & \text{if } (x_1, \ldots, x_{t(j)}) \in A^{t(j)}, \\ \langle x_1, \ldots, x_{t(j)} \rangle, & \text{otherwise.} \end{cases}$$

In addition, if $S = \omega$, then we also assume that there is a recursive function ϕ such that for all $i \in \omega$, $\phi(i)$ is an index of a Turing machine which computes R_i^A and shows that R_i^A is Γ-computable. Similarly, if $T = \omega$, then we assume that there is a recursive function ψ such that, for all $i \in \omega$, $\psi(i)$ is an index of a Turing machine which computes $\bar{f}_i^A$ and shows that $\bar{f}_i^A$ is Γ-computable.

Definition 1.2. (i) A p-time function $f : (\Sigma^*)^n \to \Sigma^*$ is *honest p-time* if there is a polynomial q such that for all $x_1, \ldots, x_n, y = f(x_1, \ldots, x_n) \Rightarrow (\forall i \leqslant n)(|x_i| \leqslant q(|y|))$.

(ii) A structure $\mathscr{A}$ as in Definition 1.1 is called an *honest p-time structure* if $\mathscr{A}$ is p-time and, for each $j \in T$, the function $\bar{f}_j^{\mathscr{A}}$ is honest p-time.

(iii) A structure $\mathscr{A}$ as in Definition 1.1 is said to have *honest witnesses* if, for any quantifier-free formula $\phi(y, x_1, \ldots, x_n)$, there is a polynomial q such that if $a_1, \ldots, a_n \in A$ and $\mathscr{A} \models (\exists y)\phi(y, a_1, \ldots, a_n)$, then there is a $z \in A$ with $|z| \leqslant q(|\langle a_1, \ldots, a_n \rangle|)$ such that $\mathscr{A} \models \phi(z, a_1, \ldots, a_n)$.

Remark 1. All honest functions are finite-to-one.

Remark 2. The coding functions defined above are all honest p-time.

For any complexity class Γ, we say that two structures $\mathscr{A}$ and $\mathscr{B}$ are Γ-isomorphic if there is an isomorphism f from $\mathscr{A}$ onto $\mathscr{B}$ and Γ-computable functions F and G such that $f = F \mid A$ (the restriction of f to A) and $f^{-1} = G \mid B$.

Definition 1.3. For any language $\mathscr{L}$ and structure $\mathscr{A}$ as defined above, the *standard extension of* $\mathscr{L}$ is $\mathscr{L} \cup \{R\}$, where R is a new unary relation symbol and the *standard extension of* $\mathscr{A}$ is the structure $\mathscr{B}$ with universe Σ^*, with $R^{\mathscr{B}} = A$, with other relations and constants agreeing with $\mathscr{A}$ and with each $f_j^{\mathscr{B}} = \bar{f}_j^A$.

The following is easily verified.

Lemma 1.1. *For any language $\mathscr{L}$ and structure $\mathscr{A}$ with standard extension $\mathscr{B}$ as defined above,*

(i) *$\mathscr{B}$ is Γ-computable if and only if $\mathscr{A}$ is Γ-computable;*

(ii) *$\mathscr{B}$ is honest if and only if $\mathscr{A}$ is honest.*

The following lemma will allow us to reduce the alphabet Σ to $\{0, 1\}$. The proof is left to the reader.

Lemma 1.2. *For any structure $\mathcal{A}$ with $A \subset \Sigma^*$, there is a structure $\mathcal{B}$ with $B \subset \{0, 1\}^*$ which is p-time isomorphic to $\mathcal{A}$. Furthermore, $\mathcal{B}$ is honest if and only if $\mathcal{A}$ is honest and $\mathcal{B}$ has honest witnesses if and only if $\mathcal{A}$ has honest witnesses.*

2. Relational structures and Boolean algebras

In this section, we consider some examples of structures which are always isomorphic to p-time structures.

A relational structure is simply a structure which has no functions. The main result in this section is Theorem 2.1, that every recursive relational structure is recursively isomorphic to a polynomial-time structure. It is important to note that the polynomial-time structure provided will have for its universe a polynomial-time set possibly different from $\{1\}^*$ or $\{0, 1\}^*$. This is an essential part of the proof, since we construct an example to show that the theorem fails if any fixed polynomial time set A is specified in advance as the universe of the structure.

A recursive Boolean algebra is a structure with two binary functions (meet and join) and one unary function (complement). Nevertheless, Theorem 2.1 can be applied to obtain a similar result, Theorem 2.6, that every recursive Boolean algebra is recursively isomorphic to a p-time Boolean algebra. We also consider the notions of honest p-time structures and of honest witnesses for Boolean algebras. The main theorem is based on the representation of a Boolean algebra as the interval algebra of a linear ordering.

Theorem 2.1. *Suppose $\mathcal{L}$ has no function symbols. Then every recursive structure $\mathcal{A} = (A, \{R_i^{\mathcal{A}}\}_{i \in S}, \{c_i^{\mathcal{A}}\}_{i \in U})$ is recursively isomorphic to a p-time structure.*

Proof. First, we can assume that A is an infinite recursive set, since otherwise the result is trivial. Let $f : \{0, 1\}^* \to A$ be a recursive bijection. Then we can define a recursive structure $\mathcal{B}$ which is isomorphic to $\mathcal{A}$ by simply defining the interpretations of the relation symbols and constant symbols so as to make f an isomorphism from $\mathcal{B}$ onto $\mathcal{A}$. Thus there is no loss of generality in assuming that $A = \{0, 1\}^*$. Now recall that S is an initial segment of the natural numbers. We will consider the case $S = \omega$. If S is finite, the proof is similar but easier. Let $\sigma_0, \sigma_1, \ldots$ be an effective enumeration of A in the usual order (first by length and then by lexicographic order.) For any $x \in \{0, 1\}^*$, we let $v(x)$ denote the number of steps needed to run the following algorithm.

"First start to list $\sigma_0, \sigma_1, \ldots$ until we find an s such that $\sigma_s = x$. Next for each $i \leq s$, list all sequences $(x_1, \ldots, x_{t(i)})$ from $\{\sigma_0, \ldots, \sigma_s\}^{t(i)}$ and compute whether $R_i(x_1, \ldots, x_{t(i)})$ holds."

Observe that the algorithm is completely uniform in x because our definition of recursive structure ensures that there is a recursive function ψ such that $\psi(i)$ is the index of a Turing machine which computes R_i. We then define a structure

$\mathcal{M} = (M, \{R_i^M\}_{i \in S}, \{c_i\}_{i \in U})$ as follows. We let $M = \{\langle x, 1^{\nu(x)}\rangle : x \in \{0,1\}^*\}$. For each $i \in S$, we define R_i^M by $R_i^M(\langle x_1, 1^{\nu(x_1)}\rangle, \ldots, \langle x_{t(i)}, 1^{\nu(x_{t(i)})}\rangle)$ to be true if and only if $R_i^{\mathcal{A}}(x_1, \ldots, x_{t(i)})$ holds. It is clear that the function $\phi(x) = \langle x, 1^{\nu(x)}\rangle$ is a recursive isomorphism form $\mathcal{A}$ onto $\mathcal{M}$.

To see that $\mathcal{M}$ is a polynomial-time structure, we need to check that B is a polynomial time set and that each relation R^M is polynomial-time.

We show that M is p-time as follows. The procedure for testing whether an input $\langle x, y\rangle$, where $y = 1^n$ is in M is the following. Simply run the algorithm described above with input x for n steps. Then $\langle x, y\rangle \in M$ if and only if the algorithm terminates in exactly n steps.

We show that the relation R_i^M is p-time as follows. First, let $R = R_i$, let $t = t(i)$ and let c be the maximum amount of time required to test $R^{\mathcal{A}}(x_1, \ldots, x_t)$ when $\{x_1, \ldots, x_t\} \subset \{\sigma_0, \sigma_1, \ldots, \sigma_{i-1}\}$. Now given input $(b_1, \ldots, b_t)$, where each $b_i = \langle x_i, 1^{\nu(x_i)}\rangle$, the procedure for testing $R^M(b_1, \ldots, b_t)$ is simply to compute $R^{\mathcal{A}}(x_1, \ldots, x_t)$. We claim that this computation takes time at most $c + \max\{|b_j| : 1 \leq j \leq t\}$. There are two cases of this claim to consider. First, if $\{x_1, \ldots, x_t\}$ is a subset of $\{\sigma_0, \ldots, \sigma_{i-1}\}$, then by the definition of c, the computation takes at most c steps. On the other hand, if at least one of the $x_j = \sigma_s$ for some $s \geq i$, then by the definition of ν, the computation takes less than $\nu(x_j)$ steps for some j; but of course $\nu(x_j) < |b_j| \leq \max\{|b_j| : 1 \leq j \leq t\}$. It follows from the proof that $\mathcal{M}$ is actually a linear time structure. This completes the proof of Theorem 2.1. □

Note that for finite structures, essentially the same result was proved independently by Grigorieff [3].

Now if we insist that the domain $M = \{0,1\}^*$ or some other fixed p-time set, then we can construct a recursive ordering which is not isomorphic to any p-time ordering on the set M. For our next result we let ω denote the order type of the non-negative integers $\{0, 1, 2, \ldots\}$, we let ω^* denote the order type of the negative integers $\{\ldots, -3, -2, -1\}$;, and we let $\omega + \omega^*$ denote the order type where the elements of ω precede all the elements of ω^*.

Theorem 2.2. *Let A be any fixed polynomial time subset of $W = \{0,1\}^*$. Then there exists a recursive linear ordering $\mathcal{L} = (W, <_W)$ of order type $\omega + \omega^*$ which is not recursively isomorphic to any polynomial time linear ordering $\mathcal{L}'$ with domain A.*

Proof. Recall that ϕ_i is the partial recursive function computed by the ith Turing machine M_i. Let $R_0, R_1, \ldots$ be an effective list of all polynomial-time binary relations on $\{0,1\}^*$. Note that for any p-time set A, any p-time relation R on $A \times A$ can be extended to a p-time relation $\bar{R}$ on $\{0,1\}^* \times \{0,1\}^*$, where

$$(x, y) \in \bar{R} \quad \Leftrightarrow \quad (x \notin A \vee y \notin A \vee (x, y) \in R).$$

For simplicity, we let (A, R_e) denote the structure with universe A and relation R which is the restriction of R_e to $A \times A$.

Let $\tau_0, \tau_1, \ldots$ be an effective enumeration of A in the usual order (first by length and then by lexicographic order).

We shall construct our desired recursive linear ordering $\mathscr{L}$ in stages. Let $\sigma_0, \sigma_1, \ldots$ be an effective listing of $\{0,1\}^*$. At any given stage s, we shall specify two sequences $a_0^s, a_1^s, \ldots, a_{n_s}^s$ and $b_0^s, b_1^s, \ldots, b_{n_s}^s$ for some $n_s \geq s$ such that $B_s = \{\sigma_0, \ldots, \sigma_{2n_s+1}\} = \{a_0^s, b_0^s, a_1^s, b_1^s, \ldots, a_{n_s}^s, b_{n_s}^s\}$. Moreover, at stage s we shall define the ordering $< = <_{\mathscr{L}}$ on $B_s \times B_s$ so that

$$a_0^s < a_1^s < \cdots < a_{n_s}^s < b_{n_s}^s < b_{n_s-1}^s < \cdots < b_0^s.$$

Our construction will ensure that for all i, $\lim_s a_i^s = a_i$ and $\lim_s b_i^s = b_i$ exist. Moreover, our construction will ensure that $\{0,1\}^* = \{a_0, b_0, a_1, b_1, \ldots\}$ and that

(a) for all i, $a_i < a_{i+1}$ and $b_{i+1} < b_i$

and

(b) for all i and j, $a_i < b_j$.

Thus $\langle \{0,1\}^*, <_{\mathscr{L}} \rangle$ will have order type $\omega + \omega^*$. To ensure that $\mathscr{L}$ is not recursively isomorphic to (A, R_e) for any e, we shall meet the following set of requirements

$P_{\langle e,k \rangle}$: There exist n and m such that one of the following four conditions holds.

(i) $\phi_k(a_n)\uparrow$ or $\phi_k(a_n) = x \notin A$.

(ii) $\phi_k(b_m)\uparrow$ or $\phi_k(b_m) = x \notin A$.

(iii) $\phi_k(a_n) = x \in A$ and there exist $n+1$ elements $v_0, \ldots, v_n$ of A such that $(v_i, x) \in R_e$ for $i = 0, \ldots, n$.

(iv) $\phi_k(b_m) = y \in A$ and there exist $m+2$ elements $w_0, \ldots, w_{m+1}$ of A such that $(w_i, y) \notin R_e$ for $i = 0, \ldots, m+1$.

We write $(w_i, y) \notin R_e$ rather than $(y, w_i) \in R_e$ in clause (iv) to allow for the possibility that R_e is not actually a linear ordering.

It is easy to see that if requirement $P_{\langle e,k \rangle}$ is satisfied, then ϕ_e is not a recursive isomorphism from $\mathscr{L} = \langle \{0,1\}^*, <_{\mathscr{L}} \rangle$ onto (A, R_e). Thus meeting all the requirements $P_{\langle e,k \rangle}$ ensures that $\mathscr{L}$ is not recursively isomorphic to any p-time linear ordering with universe A.

Our basic strategy for meeting a requirement P_z, where $z = \langle e, k \rangle$, is as follows. Let us assume that $s > z$ is a stage large enough so that requirements $P_0, \ldots, P_{z-1}$ no longer require action at any stage $t \geq s$. Then at stage s, we consider a_z^s. Our construction will then ensure that $a_j^s = a_j^t$ for all $j \leq z$ and $t \geq s$ unless there is a stage $u \geq s$ such that $\phi_k^u(a_z^s)\downarrow$. Of course if there is no such u, then $a_z^s = a_z$ and a_z will witness that requirement P_z is satisfied (by virtue of clause (i)).

Now if there is such a stage u, then let $x = \phi_k^u(a_z^s)$. If $x \notin A$, then again we will simply ensure $a_z^s = a_z$ so that once again a_z will witness that requirement P_z is satisfied. If $x \in A$, then we will compare x to the first $4n_{u-1} + 4$ elements of A (in the fixed order $\tau_0, \tau_1, \ldots$ prescribed above) with respect to the binary relation R_e. Note that since A and R_e are polynomial-time, we can effectively make these $4n_{u-1} + 4$ comparisons. There are two possibilities.

(i) There are $h = 2n_{u-1} + 2$ of these elements v of A such that $(v, x) \in R_e$ — denote these elements by $v_0, \ldots, v_{h-1}$.

(ii) There are $h = 2n_{u-1} + 3$ of these elements w of A such that $(w, x) \notin R_e$ — denote these elements by $w_0, \ldots, w_{h-1}$.

In case (i), we will simply ensure $a_z^s = a_z$. But then a_z is preceded by exactly z elements in $\mathscr{L}$, where $z \leqslant n_{u-1}$, whereas $x = \phi_k(a_z)$ is preceded by at least $2n_{u-1} + 2$ elements in (A, R_e). Thus ϕ_k is not an isomorphism from $\mathscr{L}$ onto (A, R_e).

In case (ii), we will switch $a_z^s = a_z^{u-1}$ from the ω side of $\mathscr{L}$ to the ω^* side of $\mathscr{L}$. That is, we shall let $n_u = 2n_{u-1} - z + 1$ and let $b_{n_u+i}^u = a_{n_u-i+1}$ for $i = 1, \ldots, n_u - n_{u-1}$; also let $b_i^u = b_i^{u-1}$ for $i \leqslant n_{u-1}$. Then our construction will ensure that for all $t \geqslant u$, $b_{n_u}^t = b_{n_u}^u = a_z^s$. Thus in this case, there will be precisely $n_u + 1$ elements w (namely $b_0, \ldots, b_{n_u}$) such that $(w, b_{n_u}) \notin <^L$. However, in (A, R_e) there are at least $2n_{u-1} + 3$ elements x such that $(x, \phi_k(b_{n_u})) \notin R_e$. But $n_u + 1 = 2n_{u-1} + 2 - z < 2n_{u-1} + 3$, so that ϕ_u cannot be an isomorphism from $\mathscr{L}$ onto (A, R_e). Our construction will ensure that a_z^s can switch from the ω side to the ω^*-side of $\mathscr{L}$ only for the sake of requirements $P_0, \ldots, P_z$. The usual priority argument will then show that a_z^s 'switches sides' for at most finitely many s.

We shall employ a set of movable markers Γ_e to help us keep track of which requirements we have acted on. The idea is that if we have taken an action as described above which ensures a_z^u will witness that requirement P_z is satisfied, then we will place a Γ_z marker on a_z^u. Thus at any given stage s, either Γ_z is inactive, i.e., Γ_z does not rest on any element at stage s, or Γ_z is active, i.e., Γ_z rests on some element $x \in \{a_0^s, b_0^s, \ldots, a_{n_s}^s, b_{n_s}^s\}$. If Γ_z is active, we let $\Gamma_z(s) = x$, where x is the element on which Γ_z is placed.

The Construction

Stage 0. Let $a_0^0 = \sigma_0$, $b_0^0 = \sigma_1$, and declare $a_0^0 < b_0^0$. We let Γ_z be inactive for all z at stage 0.

Stage $s + 1$. Assume we have defined $n = n_s$, $a_0^s, b_0^s, \ldots, a_n^s, b_n^s$ so that $n \geqslant s$ and

$$\{a_0^s, b_0^s, \ldots, a_n^s, b_n^s\} = \{\sigma_0, \ldots, \sigma_{2n+1}\} = B_s.$$

Moreover, assume we have defined a linear order $< = <_{\mathscr{L}}$ on $B_s \times B_s$ so that

$$a_0^s < a_1^s < \cdots < a_n^s < b_n^s < b_{n-1}^s < \cdots < b_0^s.$$

Look for a $p \leq s$ such that Γ_p is inactive at stage s and $\phi_k^s(a_p^s)\downarrow$, where $p = \langle e, k \rangle$.

If there is no such p, then for all z, let Γ_z be inactive at stage $s+1$ if and only if Γ_z is inactive at stage s; if Γ_z is active, let $\Gamma_z(s+1) = \Gamma_z(s)$. In addition, let $a_i^{s+1} = a_i^s$ and $b_i^{s+1} = b_i^s$ for all $i \leq n = n_s$. Finally, let $n_{s+1} = n+1$, $\sigma_{2n+2} = a_{n+1}^{s+1}$ and $\sigma_{2n+3} = b_{n+1}^{s+1}$ and extend our definition of $<=<_{\mathscr{L}}$ to $B_{s+1} \times B_{s+1}$ by declaring

$$a_0^{s+1} < \cdots < a_{n+1}^{s+1} < b_{n+1}^{s+1} < \cdots < b_0^{s+1}.$$

If there is such a p, let $p = p(s+1) = \langle e(s+1), k(s+1) \rangle = \langle e, k \rangle$ be the least such p and $x = x(s+1) = \phi_k(a_p^s)$. If $x(s+1) \notin A$, then proceed exactly as in the case where $p(s+1)$ is not defined, except declare Γ_p active and let $\Gamma_p(s+1) = a_p^{s+1}$. If $x(s+1) \in A$, then find the first $4n_s + 4$ elements of A. Now compare these elements to x with respect to R_e. We will then be in one of two cases.

Case 1. There are $h = 2n_s + 2$ elements $v_0, \ldots, v_{h-1}$ among the first $4n_s + 4$ elements of A such that $(v_i, x) \in R_e$ for $i = 0, 1, \ldots, h-1$. In this case, we proceed exactly as in the case where $x(s+1) \notin A$.

Case 2. Otherwise, there must be $h+1$ elements $w_0, \ldots, w_{h+1}$, among the first $4n_s + 4$ elements of A, such that $(w_i, x) \notin R_e$ for all i. Then we let $a_i^{s+1} = a_i^s$ for $i < p$ and $b_j^{s+1} = b_j^s$ for all $j \leq n = n_s$. Set $n_{s+1} = 2n + 1 - p$. Let $b_{n+i}^{s+1} = a_{n-i+1}^s$ for $i = 1, \ldots, n+1-p$ and let $a_{p+j}^{s+1} = \sigma_{2n+2+j}$ for $j = 0, \ldots, 2n+1-2p$. Activate the Γ_p marker and place it on $b_{n_s+1}^{s+1} = a_p^s$. Remove any markers Γ_z that were on elements among $a_p^s, \ldots, a_n^s$ and make them inactive. Any marker Γ_z which was active at stage s where $\Gamma_z(s) \in \{a_0^s, \ldots, a_{p-1}^s, b_0^s, \ldots, b_n^s\}$ is still active at stage $s+1$ and $\Gamma_z(s+1) = \Gamma_z(s)$. All markers Γ_z where $z \neq p$ which were inactive at stage s remain inactive at stage $s+1$. Finally, extend $<=<_{\mathscr{L}}$ to $B_{s+1} \times B_{s+1}$ by declaring

$$a_0^{s+1} < \cdots < a_{n_{s+1}}^{s+1} < b_{n_{s+1}}^{s+1} < \cdots < b_0^{s+1}.$$

This completes our construction.

Because A is a polynomial-time set and each R_e is a polynomial-time relation, it easily follows that each stage is completely effective. The following facts are easily proved by induction.

(1) For all s, $n_s \geq s$.

(2) For all s, $\{a_0^s, b_0^s, \ldots, a_{n_s}^s, b_{n_s}^s\} = \{\sigma_0, \ldots, \sigma_{2n_s+1}\}$.

(3) Our definition of $<_{\mathscr{L}}$ is consistent, that is, if $i, j \leq 2n_s + 1$ and at stage s, we declare $\sigma_i <_{\mathscr{L}} \sigma_j$, then for all $t \geq s$, we declare $\sigma_i <_{\mathscr{L}} \sigma_j$ at stage t.

Note that these facts imply that $\mathscr{L} = \langle \{0,1\}^*, < \rangle$ is a recursive linear ordering, because to decide if $\sigma_i < \sigma_j$, we simply go to stage $s = \max\{i, j\}$ and then $\sigma_i < \sigma_j$ if and only if at stage s, we declare $\sigma_i < \sigma_j$.

Next we prove two lemmas which will complete the proof that $\mathscr{L}$ has the desired properties.

Lemma 2.3. *For each z, $\lim_s a_z^s = a_z$ and $\lim_s b_z^s = b_z$ exist and there is a stage t_z such that either Γ_z is inactive at stage s for all $s \geq t_z$ or Γ_z is active at stage s and $\Gamma_z(s) = \Gamma_z(t_z)$ for all $s \geq t_z$.*

Proof. We proceed by induction on $z = \langle e, k \rangle$. By induction, we can assume that there is a stage $u > z$ large enough so that

(i) $a_j^s = a_j^u$ and $b_j^s = b_j^u$ for all $j < z$ and $s \geq u$, and

(ii) for each $j < z$, either Γ_j is inactive at stage s for all $s > u$ or for all $s \geq u$, Γ_j is active at stage s and $\Gamma_j(s) = \Gamma_j(u)$.

Note that our construction ensures that a Γ_j marker can be removed from an element at stage s only if $\Gamma_j(s-1) = a_k^s$ for some k and we take action to meet a requirement $\Gamma_{p(s)}$ at stage s where $p(s) < k$. Similarly, the only way $a_k^s \neq a_k^{s+1}$ is if $p(s+1) \leq k$ and we act according to Case 2 at stage $s+1$. Moreover, our construction ensures that if $j \leq n_s$, then $b_j^t = b_j^s$ for all $s \geq t$. It follows that $\lim_s b_z^s = b_z^u$. Now if Γ_z is active at stage s, then our choice of u ensures $p(s) > z$ for all $s > u$ so that $\lim_s a_z^s = a_z^u$ and Γ_z is active for all $s > u$. If Γ_z is not active at stage u, then either

(i) $\phi_k^s(a_z^u)\uparrow$ for all $s \geq u$, in which case, for all $s \geq u$, Γ_z is inactive at stage s, $p(s) > z$ and $a_z^s = a_z^u$, or

(ii) there is an $s > u$ such that $\phi_k^s(a_z^u)\downarrow$.

In case (ii), let t be the least $s > u$ such that $\phi_k^t(a_z^u)\downarrow$. Then our choice of u ensures that, for all $u \leq s \leq t$, $p(s) > z$ and Γ_z is inactive at stage s, so that $p(t+1)$ will be defined and $p(t+1) = z$. But then at stage $t+1$, Γ_z becomes active and is placed on either a_z^{t+1} or $b_{n_{t+1}}^{t+1}$. If Γ_z is placed on a_z^{t+1}, then $a_z^{t+1} = a_z^t$ and Γ_z will never be removed from a_z^{t+1}. This is because Γ_z can be removed from a_z^{t+1} only if $p(s) < z$ for some $s > t+1$, which is ruled out by our choice of u. If Γ_z is placed on $b_{n_{t+1}}^{t+1}$, then again Γ_z can never be removed from $b_{n_{t+1}}^{t+1}$. Thus in either case Γ_z will remain active for all $s \geq t+1$. But this means $p(s) > z$ for all $s \geq t+1$, so that $a_z^s = a_z^{s+1}$ for all $s \geq t+1$. □

Lemma 2.4. *Requirement P_p is satisfied for all p.*

Proof. Let $p = \langle e, k \rangle$ and let s_p be a stage such that $s_p > p$ and

(i) $(\forall s \geq s_p)(\forall j \leq p)[a_j^s = a_j^{s_p}$ and $b_j^s = b_j^{s_p}]$ and

(ii) $s_p \geq \max\{t_0, \ldots, t_p\}$, where t_z is a stage such that either (a) for all $s \geq t_z$, Γ_z is active at s, or (b) for all $s \geq t_z$, Γ_z is inactive at s.

It then easily follows from our construction that if Γ_p is inactive at stage s_p, then $\phi_k^s(a_p^s)\uparrow$ and $a_p^s = a_p^{s_p}$ for all $s \geq s_p$. Thus $\phi_k(a_p)\uparrow$, where $a_p = \lim_s(a_p^s)$. Hence, the requirement P_p is automatically satisfied. If Γ_p is active at stage s_p, then there are two possibilities. The first is that $\Gamma_p(s) = a_p^s = a_p$, in which case our construction guarantees that either $\phi_k^s(s_p) \notin A$ or $\phi_k^s(a_p) \in A$ but there are at least $p+1$ elements $v_0, \ldots, v_p \in A$ such that $(v_i, \phi_k(a_p)) \in R_e$ for $i = 0, \ldots, p$. The second possibility is that $\Gamma_p(s) = b_m^s = b_m$ for some $m \leq n_{s_p}$. In this case, our construction

ensures that $\phi_k^s(b_m^s) \in A$ and there are at least $m+2$ elements $w_0, \ldots, w_{m+1} \in A$ such that $(w_i, \phi_k(b_m)) \notin R_e$ for $i = 0, \ldots, m$. Thus in any case, P_p is satisfied. □

It now follows from Lemma 2.3 that $a_i <_{\mathscr{L}} a_{i+1}$ and $b_{i+1} <_{\mathscr{L}} b_i$ for all i and that $a_i <_{\mathscr{L}} b_j$ for all i and j, so that $\mathscr{L}$ is isomorphic to $\omega + \omega^*$. By our remarks preceding the construction of $\mathscr{L}$, it follows that meeting all the requirements P_p ensures that $\mathscr{L}$ is not recursively isomorphic to any polynomial time linear ordering over A.

This completes the proof of Theorem 2.2. □

Remark. The same proof works if A is any recursive set and $R_0, R_1, \ldots$ is any effective sequence of recursive binary relations. Thus, for example, the proof applies if we replace polynomial-time by primitive recursive in the statement of the theorem.

We now turn to the topic of recursive Boolean algebras.

Definition 2.1. (i) The language $\mathscr{L}$ consists of two binary function symbols $\wedge$ (meet) and $\vee$ (join), one unary function symbol $\neg$ (complement) and two constant symbols 0 (zero) and 1 (unity). A *Boolean algebra* $\mathscr{B}$ is a structure $(B, \wedge^B, \vee^B, \neg^B, 0^B, 1^B)$ for this language which satisfies the usual axioms.

(ii) Given a linear ordering $\mathscr{M} = (M, <)$ with a first element, the *interval algebra* Intalg($\mathscr{M}$) is the Boolean algebra of subsets of M generated by the left-closed, right-open intervals of M, $[a, b) = \{x: a \leqslant x < b\}$.

Lemma 2.5. *For any p-time linear ordering $\mathscr{L}$ with a first element, the interval algebra* Intalg($\mathscr{L}$) *is a p-time Boolean algebra.*

Proof. Note that, by Lemma 1.2, we may assume that L is a subset of $\{0,1\}^*$. Given $\mathscr{L}$, note that the elements of $\mathscr{A} = \text{Intalg}(\mathscr{L})$ can all be expressed in one of the following standard forms:

(1) $0_{\mathscr{A}} = [a_0, a_0)$, where a_0 is the first element of $\mathscr{L}$;

(2) $1_{\mathscr{A}} = [a_0, \infty)$, where here ∞ is a special symbol which is defined to be greater than all elements of L;

(3) $[a_1, a_2) \cup [a_3, a_4) \cup \cdots \cup [a_{2n-1}, a_{2n})$, where $a_1 <^L a_2 <^L \cdots <^L a_{2n-1}$ and either (i) $a_{2n} = \infty$ or (ii) $a_{2n} \in L$ and $a_{2n-1} < a_{2n}$.

In this way we have represented the universe A of $\mathscr{A}$ as a subset of Σ^*, where $\Sigma = \{0, 1, \infty, \cup, [,), ", "\}$. It is then easy to see that, because $\mathscr{L}$ is a p-time ordering, we can check in polynomial time whether a given string σ in Σ^* has one of the three forms above. To see that the operations of $\mathscr{A}$ are also p-time, let us analyze two of the operations, $\neg$ and $\cap$.

Given $x \in A$, it is easy to see that if x is in case (1), $\neg x = 1_A$ and if x is in case (2), $\neg x = 0_A$. If x is in case (3), then

$$\neg x = [a_0, a_1) \cup [a_2, a_3) \cup \cdots \cup [a_{2n-2}, a_{2n-1}) \cup [a_{2n}, \infty)$$

with the convention that we drop $[a_0, a_1) \cup$ from the start of $\neg x$ if $a_0 = a_1$ and we drop $\cup [a_{2n}, \infty)$ from the end of $\neg x$ if $a_{2n} = \infty$. Thus it is easy to see that the operation $\neg$ in $\mathscr{A}$ will be polynomial-time.

Next consider the operation of meet in $\mathscr{A}$. First let us consider the computing of $x \cap [b, c)$. Clearly if x is in case (1), then $x \cap [b, c) = 0_A$. If x is in case (2), then $x \cap [b, c) = [b, c)$. Now if x is in case (3), then we have four cases. Let a_0 be the first element of $\mathscr{L}$ and set $a_{2n+1} = \infty$. Then we can find i and j such that $a_i \leqslant b < a_{i+1}$ and $a_j < c \leqslant a_{j+1}$. Then it is somewhat tedious but easy to check that we have the following expression for $x \cap [b, c)$ depending on the parity of i and j. Note that we must have $b < c$, so that $i \leqslant j$.

Case 1. If $i = 2k + 1$ and $j = 2l + 1$, then

$$x \cap [b, c) = \begin{cases} [b, a_{2k+2}) \cup \cdots \cup [a_{2l+1}, c), & \text{if } k < l, \\ [b, c), & \text{if } k = l. \end{cases}$$

Case 2. If $i = 2k + 1$ and $j = 2l$, then

$$x \cap [b, c) = [b, a_{2k+2}) \cup \cdots \cup [a_{2l-1}, a_{2l}).$$

Case 3. If $i = 2k$ and $j = 2l + 1$, then

$$x \cap [b, c) = [a_{2k+1}, a_{2k+2}) \cup \cdots \cup [a_{2l+1}, c).$$

Case 4. If $i = 2k$ and $j = 2l$, then

$$x \cap [b, c) = \begin{cases} [a_{2k+1}, a_{2k+2}) \cup \cdots \cup [a_{2l-1}, a_{2l}), & \text{if } k < l, \\ 0^A, & \text{if } k = l. \end{cases}$$

Then it is easy to see that if $y = [b_1, b_2) \cup \cdots \cup [b_{2m-1}, b_{2m})$, then

$$x \cap y = (x \cap [b_1, b_2)) \cup \cdots \cup (x \cap [b_{2m-1}, b_{2m})),$$

where we make the convention that if $(x \cap [b_{2i-1}, b_{2i})) = 0^A$ then we delete the string $(x \cap [b_{2i-1}, b_{2i}))$ and the appropriate occurrence of $\cup$ from the expression. Of course there is one further special case: if $(x \cap [b_{2i-1}, b_{2i})) = 0^A$ for each i, then we set $x \cap y = 0^A$. Now it is easy to see that the meet operation in $\mathscr{A}$ is p-time. We need only compare the elements of L in the two expressions, and these comparisons are polynomial-time in the length of the expressions because the ordering $<^L$ is assumed to be a polynomial-time linear ordering. A similar type of argument will show that the join operation is also p-time. □

Theorem 2.6. *Every recursive Boolean algebra $\mathscr{B}$ is recursively isomorphic to a p-time Boolean algebra.*

Proof. First observe that the classical proof that every countable Boolean algebra is isomorphic to the interval algebra of a countable linear ordering is effective. (See Remmel [6].) Thus every recursive Boolean algebra is recursively isomorphic to Intalg($\mathcal{M}$), where $\mathcal{M}$ is a recursive linear ordering. The result now follows from Lemma 2.5 and Theorem 2.1. □

The next theorem demonstrates that this result cannot be improved from p-time to honest p-time or even to p-time with honest witnesses.

Theorem 2.7. (a) *Every honest p-time Boolean algebra is finite.*

(b) *Every recursive Boolean algebra with honest witnesses has a recursive set of atoms.*

(c) *There exists a p-time Boolean algebra which is not isomorphic to any p-time Boolean algebra with honest witnesses.*

Proof. (a) Note that if $\mathcal{B} = (B, \wedge, \vee, \neg, 0^B, 1^B)$ is an honest p-time Boolean algebra, then $\wedge$ is a finite-to-one function. But for each $b \in B$, $b \wedge \neg b = 0^B$. Thus B must be finite.

(b) Let $\mathcal{B}$ be a p-time Boolean algebra with honest witnesses. Note that for any $b \in B$,

$$b \text{ is not an atom} \Leftrightarrow (\exists y)[y \wedge b \neq 0^B \,\&\, \neg y \wedge b \neq 0^B].$$

Clearly this is decidable if $\mathcal{B}$ has honest witnesses.

(c) It is a theorem of Goncharov [2], that there exists a recursive atomic Boolean algebra $\mathcal{B}$ which is not isomorphic to any recursive Boolean algebra $\mathcal{D}$ such that the set of atoms of $\mathcal{D}$ is recursive. By Theorem 2.6, $\mathcal{B}$ is isomorphic to a p-time Boolean algebra $\mathcal{A}$. It follows from part (b) that $\mathcal{A}$ is a p-time Boolean algebra which is not isomorphic to any p-time structure with honest witnesses. □

The following result shows that the concept of honest witnesses for Boolean algebras is non-trivial.

Theorem 2.8. *There exists an infinite polynomial-time Boolean algebra with honest witnesses.*

Proof. We first construct a countable atomless Boolean algebra which is p-time and has honest witnesses. We then indicate how a slight variation of the construction yields such a Boolean algebra within infinitely many different classical isomorphism types.

Construction of a polynomial time atomless Boolean algebra with honest witnesses

We shall give a construction of a p-time Boolean algebra which first appeared in Nerode and Remmel [5].

Let $\mathcal{M}$ be the linear ordering of the set of dyadic rationals $q = i/2^n$ with $0 \leq q < 1$, where the rational 0 is represented by the string (0) and each dyadic rational $q > 0$ is represented by a string $i_1 i_2 \cdots i_n \in \{0, 1\}^*$, where $i_n = 1$ and $q = \sum_j i_j 2^{-j}$. Thus, for example, 1011 represents $2^{-1} + 2^{-3} + 2^{-4} = 11/16$. It is clear that $\mathcal{M}$ is a p-time linear ordering.

Now let $\mathcal{B} = \text{Intalg}(\mathcal{M})$. It follows from Lemma 2.3 that $\mathcal{B}$ is a p-time Boolean algebra.

It remains to be shown that $\mathcal{B}$ has honest witnesses.

First we make the following general remarks about Boolean algebras.

For any subset S of $\mathcal{B}$, let S^* denote the subalgebra generated by S and let $I(S)$ denote the ideal generated by S. Given a subalgebra $\mathcal{D}$ of $\mathcal{B}$, we let $\text{At}(\mathcal{D})$ denote the set of atoms of $\mathcal{D}$.

Suppose that $\phi(y, x_1, \ldots, x_n)$ is a quantifier-free formula and that $\mathcal{B} \models (\exists y)\phi(y, d_1, \ldots, d_n)$. Let $c \in B$ with $\mathcal{B} \models \phi(c, d_1, \ldots, d_n)$. Let $\mathcal{D} = \{d_1, \ldots, d_n\}^*$ and let $\{a_1, \ldots, a_k\} = \text{At}(\mathcal{D})$.

Now suppose $b \in B$ and

$$(*) \qquad \text{for all } a \in \text{At}(\mathcal{D}) \quad \begin{cases} b \wedge a = 0 & \Leftrightarrow \quad c \wedge a = 0, \\ \neg b \wedge a = 0 & \Leftrightarrow \quad \neg c \wedge a = 0. \end{cases}$$

Then there is an isomorphism

$$f : \{b, d_1, \ldots, d_n\}^* \to \{c, d_1, \ldots, d_n\}^*,$$

where $f(a) = a$ for $a \in \text{At}(\mathcal{A})$ (and hence $f(d_i) = d_i$ for $i = 1, \ldots, n$) and $f(b) = c$.

Thus if b satisfies (*), our isomorphism shows that $b, d_1, \ldots, d_n$ and $c, d_1, \ldots, d_n$ satisfy the same quantifier-free formulas, so that

$$\mathcal{B} \models \phi(b, d_1, \ldots, d_n) \quad \Leftrightarrow \quad \mathcal{B} \models \phi(c, d_1, \ldots, d_n).$$

Thus the problem of finding honest witnesses for elements $d_1, \ldots, d_n$ of $\mathcal{B}$ reduces to two problems:

(1) Finding the atoms a of $\mathcal{D} = \{d_1, \ldots, d_n\}^*$.

(2) Finding elements b which satisfy every possible conjunction of $b \wedge a = 0$ (or $b \wedge a \neq 0$) and $\neg b \wedge a = 0$ (or $b \wedge a \neq 0$) where a ranges over $\text{At}(\mathcal{D})$.

To solve these problems we need to refine the structure of $\mathcal{D}$. Let

$$Q(D) = \{q_0, q_1, q_2, \ldots, q_k\}$$

list in increasing order the elements 0 and ∞ together with those dyadic rationals which appear as end-points in the intervals of the elements of $\mathcal{D}$.

Thus

$$0 = q_0 < q_1 < \cdots < q_{k-1} < q_k = \infty.$$

Note that $k < |q_0| + \cdots + |q_k| < |\langle d_1, \ldots, d_n \rangle|$.

Now $\mathcal{D} = \text{Intalg}(Q(D))$ and it is clear that $\mathcal{D}$ has exactly k atoms, namely, the k

disjoint intervals

$$a_0 = [q_0, q_1), \ldots, a_k = [q_{k-1}, q_k).$$

For each $i < k$, choose a dyadic rational r_i of minimal length such that $q_i < r_i < q_{i+1}$. Note that $|r_i| \leq 1 + \max\{|q_i|, |q_{i+1}|\}$.

Let

$$Q^+(D) = Q(D) \cup \{r_0, \ldots, r_{k-1}\}$$

and let

$$\mathcal{D}^+ = \text{Intalg}(Q^+(D)).$$

The following observation is essential. $\mathcal{D}^+$ has $2k$ atoms and each atom $a_i = [q_i, q_{i+1})$ of $\mathcal{D}$ is split into two atoms $b_{2i} = [q_i, r_i)$ and $b_{2i+1} = [r_i, q_{+1})$ of $\mathcal{D}$.

Now recall the formula ϕ and the element c of B with $\phi(c, d_1, \ldots, d_n)$. We claim that there is an element b of $\mathcal{D}^+$ such that (*) is satisfied.

The element b is defined as follows. For each $i < k$, let c_i be defined in three cases.

Case 1. If $c \wedge a_i \neq 0$ and $\neg c \wedge a_i \neq 0$, let $c_i = b_{2i}$.

Case 2. If $c \wedge a_i \neq 0$ and $\neg c \wedge a_i = 0$, let $c_i = a_i$.

Case 3. If $c \wedge a_i = 0$ and $\neg c \wedge a_i \neq 0$, let $c_i = 0$.

Note that since $a_i \neq 0$, it is not possible that $c \wedge a_i = 0 = \neg c \wedge a_i$.

Now let $b = \bigcup_i c_i$. It is clear that, for each i,

$$b \wedge a_i = 0 \quad \Leftrightarrow \quad c_i \wedge a_i = 0 \quad \Leftrightarrow \quad c \wedge a_i = 0$$

and

$$\neg b \wedge a_i = 0 \quad \Leftrightarrow \quad \neg c_i \wedge a_i = 0 \quad \Leftrightarrow \quad \neg c \wedge a_i = 0.$$

It follows from the discussion above that $\mathcal{B} \models \phi(b, d_1, \ldots, d_n)$.

It remains to calculate an upper bound for the length of b. Now b is the union of some subset S of the atoms $\{b_0, b_1, \ldots, b_{2k-1}\}$ of $\mathcal{D}^+$, where S never contains two consecutive atoms. Thus b includes at most k intervals and each rational from $Q^+(D)$ occurs at most once as an end-point. It follows that

$$|b| \leq 4k + 3 + |q_0| + \cdots + |q_k| + |r_0| + \cdots + |r_{k-1}|.$$

Now for each i, either $|r_i| \leq |q_i| + 1$ or $|r_i| \leq |q_{i+1}| + 1$. In either case, $|r_i| \leq |q_i| + |q_{i+1}|$. It follows that

$$|r_0| + \cdots + |r_{k-1}| \leq |q_0| + 2|q_1| + \cdots + 2|q_{k-1}| + |q_k|.$$

Since $|q_0| = |q_k| = 1$, we now have

$$|b| \leq 4k + 7 + 3(|q_1| + \cdots + |q_{k-1}|).$$

On the other hand, each of the rationals $q_1, \ldots, q_{k-1}$ occurs as an endpoint in some d_j, so that the $d_1, \ldots, d_n$ collectively include at least $(k-1)/2$ intervals and

we have

$$|\langle d_1, \ldots, d_n\rangle| > |q_1| + \cdots + |q_{k-1}| + 3(k-1)/2.$$

Thus we see that

$$|b| \leqslant 3\,|\langle d_1, \ldots, d_n\rangle| + 12.$$

It follows that b is an honest witness. This demonstrates that the Boolean algebra $\mathscr{B}$ defined above has honest witnesses and completes the proof of Theorem 2.8. □

To get infinitely many isomorphism types of p-time Boolean algebras with honest witnesses, one can simply modify the construction of $\mathscr{B}$ by changing the order $\mathscr{M}$. For example, if we let $\mathscr{M} = \{0\} \cup \{2^{-n} : n \in \mathbb{N}\}$ with the usual ordering as a subset of the rationals, then $\mathscr{B}$ will turn out to be isomorphic to the Boolean algebra of finite and cofinite subsets of ω.

3. Structures with functions

In this section we consider recursive structures which are not recursively isomorphic to polynomial-time structures. In fact, we show that recursive structures can be distinguished from primitive recursive structures and exponential-time structures can be distinguished from polynomial-time structures. This can be done with just one unary function.

Theorem 3.1. *Let $\mathscr{L}_0$ be the language which has no relation symbols and no constant symbols and has exactly one function symbol f which is unary.*

(i) *There is a recursive structure $\mathscr{A} = (A, f^{\mathscr{A}})$ which is not recursively isomorphic to any primitive recursive structure.*

(ii) *There is an exponential-time structure $\mathscr{D} = (D, f^{\mathscr{D}})$ which is not recursively isomorphic to any polynomial-time structure.*

Proof. (i) Let (A_0, f_0), (A_1, f_1), ... be an effective list of all primitive recursive structures over $\mathscr{L}_0$ and let ϕ_0, ϕ_1, ... be a list of all one-to-one partial recursive functions. We must meet the following set of requirements in our construction of $\mathscr{A}$:

$$R_{i,j}: \quad \phi_j \text{ is not a recursive isomorphism from } \mathscr{A} \text{ to } (A_i, f_i).$$

To meet the requirements $R_{i,j}$, recursively partition $\{0, 1\}^*$ into infinitely many disjoint infinite recursive sets $S_{i,j}$. We then define $\mathscr{A} = (A, f^{\mathscr{A}})$ so that $A = \bigcup_{i,j} S_{i,j} = \{0, 1\}^*$ and for all i, j, f maps $S_{i,j}$ into $S_{i,j}$.

We now fix i, j and then we define $f = f^{\mathscr{A}}$ on $S_{i,j}$ in stages. We let $a_0, a_1, \ldots$ be some effective listing of $S_{i,j}$. At stage s, we shall define $f(a_s)$. We start by defining

$f(a_0) = a_1$ at stage 0. At stage $s+1$, compute $\phi_j^s(a_0)$. If $\phi_j^s(a_0)\uparrow$ or if $\phi_j^{s-1}(a_0)\downarrow$, then we define $f(a_s) = a_{s+1}$. Otherwise, that is, if $\phi_j^s(a_0)\downarrow$ but $\phi_j^{s-1}(a_0)\uparrow$, let $x = \phi_j^s(a_0)$ and do the following. Compute the sequence $x, f_i(x), f_i(f_i(x)), \ldots, f_i^{(s+1)}(x)$, where here $f_i^{(k)}$ denotes f composed with itself k times. Note that if ϕ_j were an isomorphism, then it must be the case that $\phi_j(a_k) = f_i^{(k)}(x)$ for all k. Thus if ϕ_j were an isomorphism, then it must be the case that

$$f(a_s) = a_0 \quad \Leftrightarrow \quad f_i^{(s+1)}(x) = x.$$

Thus if $f_i^{(s+1)}(x) = x$, then we define $f(a_s) = a_{s+1}$. If $f_i^{(s+1)}(x) \neq x$, then we define $f(a_s) = a_0$. Note that in either case, we will have ensured that ϕ_j cannot be an isomorphism from $\mathcal{A}$ onto (A_i, f_i).

This completes the proof of part (i).

(ii) The proof of part (i) must be modified in several ways. First, let (E_0, f_0), $(E_1, f_1), \ldots$ be an effective list of all p-time structures over $\mathcal{L}_0$; again let $\phi_0, \phi_1, \ldots$ be a list of all one-to-one partial recursive functions. We shall build our structure $(D, f^{\mathcal{D}})$ so that $D \subset \omega = \{1\}^*$, the natural numbers in unary form. For the rest of this proof, all natural numbers are assumed to be given in unary form.

We must meet the following set of requirements in our construction of $\mathcal{A}$:

$$R_{i,j}: \quad \phi_j \text{ is not a recursive isomorphism from } \mathcal{D} \text{ to } (E_i, f_i).$$

To meet the requirements $R_{i,j}$, we construct D as a disjoint polynomial-time union of infinite p-time sets $D = \bigcup_{i,j} T_{i,j}$.

Define the usual coding function $[\cdot, \cdot]$ from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$ by

$$[i, j] = \tfrac{1}{2}(i+j)^2 + 3i + j + 1.$$

Notice that $x = [i, j]$ can be computed from input (i, j) in time $a \cdot x$ for some fixed constant a.

Now define the function $\psi : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ by the following recursion.

$$\psi(0, i, j) = 2[i, j] + 1, \qquad \psi(n+1, i, j) = 2^{\psi(n,i,j)}.$$

Note that the computation of $y = 2^x$ from input x again takes time at most $b \cdot y$ for some fixed constant $b \geqslant a$. Thus the computation of $z = \psi(n, i, j)$ from input (n, i, j) takes at most the following number of steps:

$$b[i, j] + bx + b2^x + b2^{2^x} + \cdots + bz < b(1 + 2 + \cdots + z) < bz^2.$$

For each i, j, let $T_{i,j} = \{\psi(n, i, j) : n < \omega\}$.

Now let $R(z, n, i, j) \Leftrightarrow z = \psi(n, i, j)$. Then to test $R(z, n, i, j)$, perform the computation of $\psi(n, i, j)$ for bz^2 steps — $R(z, n, i, j)$ holds if the computation converges to z by that time. Thus R is p-time. It follows that the sets $T_{i,j}$ are uniformly p-time and that D is also p-time, since

$$z \in T_{i,j} \quad \Leftrightarrow \quad (\exists n < z)\, R(y, n, i, j)$$

and

$$z \in D \quad \Leftrightarrow \quad (\exists i, j < z)\, z \in T_{i,j}.$$

We now fix i, j and define $f = f^{\mathscr{D}}$ on $T_{i,j} = \{a_0, a_1, \ldots\}$, where $a_n = \psi(n, i, j)$.

For each m, perform the following series of computations in time bounded by 2^{a_m}.

(1) Start to compute $\phi_j(a_0)$. If this converges in time $< 2^{a_m}$, let $b_0 = \phi_j(a_0)$.

(2) Check that $b_0 \in E_i$.

(3) Compute the sequence $b_1 = f_i(b_0)$, $b_2 = f_i(b_1)$, $\ldots$, $b_{m+1} = f_i(b_m)$.

Let s be the least m such that the computations can be successfully completed. Let us demonstrate that, assuming the existence of $b_0 = \phi_j(a_0)$, such an m must exist. (Once again, if $\phi_j(a_0)$ diverges, then condition $R_{i,j}$ is automatically satisfied.)

Now it takes some constant amount c_0 of time to compute b_0. Since f_i is p-time, $f_i(y)$ can be computed in time bounded by $|y|^k$ for some fixed integer $k > 1$ and any $y \in \{0, 1\}^*$ with $|y| > 1$. Let c_1 be the time required to compute $f_i(\emptyset)$, $f_i(0)$ and $f_i(1)$, if needed, and let $c = c_0 + c_1$.

Then to compute the sequence $b_0 = \phi_j(a_0)$, $b_1 = f_i(b_0)$, $\ldots$, $b_{m+1} = f_i(b_m)$ takes at most

$$\begin{aligned} t(m) &= c + |b_0|^k + (|b_0|^k)^k + \cdots \\ &= c + |b_0|^k + |b_0|^{k^2} + \cdots + |b_0|^{k^m}. \end{aligned}$$

We need to show that this sequence is eventually dominated by the sequence $a_0, a_1 = 2^{a_0}, \ldots, a_m = 2^{a_{m-1}}$.

We may assume without loss of generality that $|b_0| > 1$. Now if m is large enough so that both c and m are $< |b_0|^{k^m}$, then

$$t(m) < (m+1)\,|b_0|^{k^m} \leqslant |b_0|^{2k^m}.$$

Now let m be large enough so that $|b_0|^2 < 2^{2^m}$ and $k < 2^m$, and so that $m^2 + m < 2^m$. Then $k^m < 2^{m^2}$ and

$$t(m) < 2^{2^m \cdot 2^{m^2}} = 2^{2^{m^2+m}} < 2^{2^{2^m}} = \exp_3(m).$$

To show that the latter is dominated by a_m, note first that $a_0 \geqslant 3$ and that, for any m, $a_m \geqslant m + 3$; it follows that $a_{m+3} \geqslant \exp_3(m+3)$.

The definition of f now proceeds in stages, as in part (i). Let s be the least m such that the computations described above can be successfully completed. Now for $t \neq s$, we let $f(a_t) = a_{t+1}$. To compute $f(a_s)$, we let $x = \phi_j(a_0)$ and compute $f_i^{(s+1)}(x)$. Then if $f_i^{(s+1)}(x) = x$, we define $f(a_s) = a_{s+1}$. If $f_i^{(s+1)}(x) \neq x$, then we define $f(a_s) = a_0$. Note that in either case, we will have ensured that ϕ_j cannot be an isomorphism from $\mathscr{D}$ onto (A_i, f_i).

It remains to be seen that the computation of f can be done in exponential time. Given $x \in D$, we first compute the unique triple (n, i, j) such that $x = \psi(n, i, j)$; this can be done in polynomial time since n, i and j are all less than

x. Next we perform the computations (1), (2) and (3) for $m = 0, 1, \ldots, n$ in turn. This can be done in exponential time since each series of computations is bounded by time 2^{a_m}. The remainder of the computation of $f(x)$ takes little time. We look for the least n, if any, such that the nth series of computations has been successfully completed. If $m = n$, then we check to see if $f_i^{(s+1)}(x) = x$ and let $f(x) = 2^x$ if so; otherwise, $f(x) = a_0 = [i, j]$.

This completes the proof of part (ii). □

By adding a second unary function, we can improve on Theorem 3.1.

Lemma 3.2. *For any Δ_2^0 function ϕ, there is a uniformly p-time sequence ϕ_t, $t = 0, 1, 2, \ldots$, such that $\phi = \lim_t \phi_t$.*

Proof. Let ϕ be Δ_2^0. By the Limit Lemma, there is a uniformly recursive sequence $\psi_0, \psi_1, \ldots$ such that $\phi = \lim_n \psi_n$. For simplicity, assume that ψ_0 is the constant 0 function and takes time $t = 1$ to compute. For each n, t and x, let $n(t)$ be the largest $n \leq t$ such that $\psi_n(x)$ converges in time $\leq t + 1$ and let

$$\phi_t(x) = \psi_{n(t)}(x).$$

It is clear that the computation of $\phi_t(x)$ takes time at most $(t + 1)^2$. □

Theorem 3.3. *Let $\mathscr{L}_1$ be the language which consists of two unary function symbols f, g.*

(i) *There is a recursive structure $\mathscr{A} = (A, f^A, g^A)$ over $\mathscr{L}_1$ which is not Δ_2^0-isomorphic to any primitive recursive structure.*

(ii) *There is an exponential-time structure $\mathscr{D} = (D, f^D, g^D)$ over $\mathscr{L}_1$ which is not Δ_2^0-isomorphic to any polynomial-time structure.*

Proof. (i) Let (A_0, f_0, g_0), $(A_1, f_1, g_1), \ldots$ be an effective list of all primitive recursive structures over $\mathscr{L}_1$. Let ϕ_i be the partial recursive function computed by the ith Turing machine and let $\psi_i = \lim_t \phi_{\phi_i(t)}$. We say $\lim_t \phi_{\phi_i(t)}$ exists if and only if, for each t, $\phi_{\phi_i(t)}$ is total and, for each x, $\lim_t \phi_{\phi_t}(x)$ exists.

We shall construct a recursive $\mathscr{L}_1$ structure (A, f, g) such that we meet the following set of requirements $R_{i,j}$: If $\lim_t \phi_{\phi_i(t)} = \psi_i$ exists, then ψ_i is not an isomorphism from (A, f, g) onto (A_j, f_j, g_j).

As in the proof of Theorem 3.1, we define infinitely many pairwise disjoint p-time subsets $S_{i,j}$ of $\{0, 1\}^*$ and define f and g on $S_{i,j}$ to ensure that requirement $R_{i,j}$ is met. Specifically, for each $i, j, k, n \in \mathbb{N}$, we let $a_{i,j,k,n} = 0^i 1 0^j 1 0^{k+1} 1^{\psi(n,i,j)}$, where $\psi(n, i, j)$ is as defined in the proof of Theorem 3.1. We then let $S_{i,j} = \{a_{i,j,k,n} : k, n \in \mathbb{N}\}$. We let $A = \bigcup_{i,j} S_{i,j}$. It is easy to see from our analysis of the function $\psi(n, i, j)$ in Theorem 3.1 that A is a polynomial time set.

Our strategy for meeting a single requirement $R_{i,j}$ is very similar to our strategy in Theorem 3.1. First we shall define g on all of $A = \bigcup_{i,j} S_{i,j}$ by setting

$g(a_{i,j,k,n}) = a_{i,j,k+1,n}$. It is easy to see that g is the restriction to A of a polynomial-time function. For each fixed i, j, k, we shall define f on the sequence $a_{i,j,k,0}, a_{i,j,k,1}, \ldots$ in much the same manner as we defined f on the sequence $a_0, a_1, \ldots$ in the construction of Theorem 3.1. That is, suppose $\psi_i = \lim_t \phi_{\phi_i(t)}$ were an isomorphism from (A, f, g) onto (A, f_j, g_j) and $\psi_i(a_{i,j,0,0}) = x_0 = \phi_{\phi_i(k)}(a_{i,j,0,0})$. Moreover, suppose we have defined $f(a_{i,j,k,t})$ for $t \leq s$. Then because $g^{(k)}(a_{i,j,0,0}) = a_{i,j,k,0}$, we have $f^{(s)}(g^{(k)}(a_{i,j,0,0})) = a_{i,j,k,s}$. Thus

$$f(a_{i,j,k,s}) = a_{i,j,k,s} \Leftrightarrow f(f^{(s)}(g^{(k)}(a_{i,j,0,0}))) = f^{(s)}(g^{(k)}(a_{i,j,0,0}))$$
$$\Leftrightarrow f_j(f_j^{(s)}(g_j^{(k)}(x_0))) = f_j^{(s)}(g_j^{(k)}(x_0)).$$

Thus we define $f(a_{i,j,k,s})$ as follows. First we compute s steps of the following computation:

$C_{i,j,k}$: "First compute $\phi_i(k)$ and if $\phi_i(k)\downarrow$, then compute $\phi_{\phi_i(k)}(a_{i,j,0,0})$".

If we complete the computation $C_{i,j,k}$ in exactly s steps, then let $x_0 = \phi_{\phi_i(k)}(a_{i,j,0,0})$. If $x_0 \in A_0$, then compute $y_0 = f_j^{(s)}(g_j^{(k)}(x_0))$. Then if $f_j(y_0) = y_0$, define $f(a_{i,j,k,s}) = a_{i,j,k,s+1}$ and if $f_j(y_0) \neq y_0$, define $f(a_{i,j,k,s}) = a_{i,j,k,s}$. Finally, if the computation $C_{i,j,k}$ is not completed in exactly s steps, or if $x_0 \notin A$, then let $f(a_{i,j,k,s}) = a_{i,j,k,s+1}$.

Note that our definition is not consistent with ψ_i being an isomorphism from (A, f, g) onto (A_j, f_j, g_j) and $\psi_i(a_{i,j,0,0}) = \phi_{\phi_i(k)}(a_{i,j,0,0})$. Thus we must conclude that either ψ_i is not an isomorphism from (A, f, g) onto (A_j, f_j, g_j) or there is no k such that $\psi_i(a_{i,j,0,0}) = \phi_{\phi_i(k)}(a_{i,j,0,0})$. In the latter case, $\phi_i(a_{i,j,0,0}) \neq \lim_t \phi_{\phi_i(t)}(a_{i,j,0,0})$, so that $\lim_t \phi_{\phi_i(t)}$ does not exist and hence requirement $R_{i,j}$ is automatically satisfied. Thus in either case our definition of f on $S_{i,j}$ ensures that requirement $R_{i,j}$ is satisfied.

Note that it is easy to see that our definition of f ensures that f is a partial recursive function from A into A. Thus (A, f, g) is a recursive $\mathscr{L}_1$ structure. Since every Δ_2^0 function is equal to ψ_i for some i, meeting all the requirements $R_{i,j}$ ensures that (A, f, g) is not Δ_2^0-isomorphic to any primitive recursive $\mathscr{L}_1$-structure.

(ii) We only have to make minor modifications in the above construction. First we let (A_0, f_0, g_0), $(A_1, f_1, g_1), \ldots$ be an effective list of all polynomial-time recursive structures over $\mathscr{L}_1$. Second, we change the definition of f slightly. We define $f(a_{i,j,k,s})$ as follows. Perform the following series of computations in time bounded by $2^{a_{i,j,k,s}}$:

(1) Compute $\phi_i(k)$.

(2) If $\phi_i(k)$ converges, compute $\phi_{\phi_i(k)}(a_{i,j,0,0})$.

(3) If $\phi_{\phi_i(k)}(a_{i,j,0,0})$ converges, let $x_0 = \phi_{\phi_i(k)}(a_{i,j,0,0})$ and check whether $x_0 \in A_j$.

(4) If $x_0 \in A_j$, compute the sequence $x_0, x_1 = g_j(x_0), \ldots, x_k = g_j(x_{k-1})$ and the sequence $y_1 = f_j(x_k), y_2 = f_j(y_1), \ldots, y_{s+1} = f_j(y_s)$.

Then if the entire sequence of computations cannot be completed in $2^{a_{i,j,k,s}}$ steps, define $f(a_{i,j,k,s}) = a_{i,j,k,s+1}$. If the entire sequence of computations is completed in $2^{a_{i,j,k,s}}$ steps, define $f(a_{i,j,k,s}) = a_{i,j,k,s}$ if $y_{s+1} \neq f_j(y_s)$ and $f(a_{i,j,k,s}) = a_{i,j,k,s+1}$ if $y_{s+1} = f_j(y_s)$.

Then we can argue exactly as in Theorem 3.1 that for each i, j, and k, the fact that A_j, f_j, and g_j are polynomial-time ensures that there will be an s such that the sequence of computations (1)–(4) for $a_{i,j,k,s}$ will be completed in $2^{a_{i,j,k,s}}$ steps. It will then follow by the same argument as in part (i) that our requirement $R_{i,j}$ is satisfied. Moreover, it is easy to see that our definition of f ensures that f is restriction to A of an exponential-time function. Thus (A, f, g) will be an exponential-time structure which is not Δ_2^0-isomorphic to any polynomial-time structure. □

Theorem 3.4. *Let $\mathscr{L}_2$ be the language which consists of one binary function symbol h.*

(i) *There is a recursive structure $\mathscr{B} = (B, h^B)$ over $\mathscr{L}_2$ which is not Δ_2^0-isomorphic to any primitive recursive structure.*

(ii) *There is an exponential-time structure $\mathscr{D} = (D, h^D)$ over $\mathscr{L}_2$ which is not Δ_2^0-isomorphic to any polynomial-time structure.*

Proof. This follows directly from Theorem 3.3. We give the proof for part (i); the proof of part (ii) is similar. Let $\mathscr{A} = (A, f, g)$ be the recursive structure over $\mathscr{L}_1$ which is not Δ_2^0-isomorphic to any primitive recursive structure. Let $B = A \cup \{c, d\}$ and define $h = h^B$ as follows. For any $a \in A$, $h(a, c) = f(a)$ and $h(a, d) = g(a)$; for any other pair $(x, y) \in B \times B$, $h(x, y) = c$.

Now suppose that there was a Δ_2^0-isomorphism ϕ mapping $\mathscr{B}$ to some primitive recursive structure $\mathscr{C} = (C, h^C)$. Let $c_0 = \phi(c)$, $d_0 = \phi(d)$ and let $M = C \setminus \{c_0, d_0\}$. Define f^M and g^M by $f^M(x) = h^C(x, c_0)$ and $g^M(x) = h^C(x, d_0)$. It follows that $\mathscr{M}$ is a primitive recursive structure. Now ϕ maps $A = B \setminus \{c, d\}$ onto $M = C \setminus \{c_0, d_0\}$ and, for each $a \in A$,

$$\phi(f(a)) = \phi(h(a, c)) = h^C(\phi(a), \phi(c)) = f^M(\phi(a)).$$

Thus ϕ is a Δ_2^0-isomorphism from $\mathscr{A}$ onto the primitive recursive structure $\mathscr{M}$. This contradiction completes the proof. □

Next we show that the structure of Abelian groups can be built into the models of Theorem 3.1.

We need the following definition and lemma.

Definition 3.1. (a) The language $\mathscr{L}$ of (commutative) group theory consists of a single binary function symbol f (addition) and a single constant symbol e (the identity). For any sequence $\mathscr{A}_n = (A_i, f_i, e_i)$ of structures for $\mathscr{L}$, the *direct sum* $\mathscr{A} = \sum_n \mathscr{A}_n$ is defined to have domain $A = \{\langle \sigma_1, \sigma_2, \ldots, \sigma_k \rangle : k \in \omega, \sigma_i \in A_i$ for $1 \leq i \leq k$ and $\sigma_k \neq e_k\}$, identity $e^A = \emptyset$ and addition f^A defined as follows: for $\sigma = \langle \sigma_1, \sigma_2, \ldots, \sigma_m \rangle$ and $\tau = \langle \tau_1, \tau_2, \ldots, \tau_n \rangle$, $f^A(\sigma, \tau) = \rho = \langle \rho_1, \rho_2, \ldots, \rho_k \rangle$, where $k = \max\{i: (i \leq m \wedge i \leq n \wedge f_i(\sigma_i, \tau_i) \neq e_i) \vee m < i \leq n \vee$

$n < i \leq m\}$ and, for $i \leq k$,

$$\rho_i = \begin{cases} f_i(\sigma_i, \tau_i), & \text{for } i \leq \min(m, n), \\ \sigma_i, & \text{for } n < i \leq k, \\ \tau_i, & \text{for } m < i \leq k. \end{cases}$$

Lemma 3.5. *Let Γ be one of the following complexity classes: recursive, primitive recursive, exponential-time, polynomial-time. Suppose that the sequence $\mathcal{A}_n = (A_i, f_i, e_i)$ of Abelian groups is uniformly Γ-computable. Then the direct sum of the sequence is also Γ-computable.*

Proof. This follows from the fact that the coding and uncoding functions are all computable in linear time. □

Theorem 3.6. (i) *There is a recursive Abelian group G which is not recursively isomorphic to any primitive recursive Abelian group.*

(ii) *There is an exponential-time Abelian group which is not recursively isomorphic to any polynomial-time Abelian group.*

Proof. (i) Let (A_0, f_0), (A_1, f_1), ... be an effective list of all primitive recursive Abelian groups, where f_i is a binary function which gives the group operation on A_i and let $\phi_0, \phi_1, \ldots$ be a list of all one-to-one partial recursive functions as before.

Let S be the infinite recursive set $\{a_n : n \in \mathbb{Z}\}$, where $a_n = \exp_3(n) = 2^{2^{2^n}}$ for $n \geq 0$ and $a_{-n} = a_n + 1$ for $n > 0$. For each i, j, we will define a set $C_{i,j}$ which is either S or a finite subset. We will then define a group structure $\mathcal{C}_{i,j}$ on each $C_{i,j}$ and let $\mathcal{A} = \sum_n \mathcal{B}_n$, where, for each n, $\mathcal{B}_n = \mathcal{C}_{(n)_1,(n)_2}$.

If ϕ_j were a group isomorphism of $\mathcal{A}$ onto $\mathcal{A}_i$, then ϕ_j' would be an embedding of $\mathcal{C}_{i,j} = \mathcal{B}_n$ into $\mathcal{A}_i$, where $\phi_j'(a) = \phi_j(\langle e_0, \ldots, e_{n-1}, a\rangle)$. Therefore we will meet the following set of requirements.

$R_{i,j}$: ϕ_j' is not a group isomorphism of $C_{i,j}$ into $\mathcal{A}_i$.

Fix i, j, let a_0 be the identity of $\mathcal{C}_{i,j}$ and define the group addition $f = f_{i,j}$ on $\mathcal{C}_{i,j} = \{a_0, a_1, a_{-1}, \ldots\}$ (where $a_0 = e_{i,j}$), as follows.

For $k = -1, 0, 1$, let $x_k = \phi_j'(a_k)$ and let s be the total number of steps required to compute all three — thus $s > 2$. If the computations do not all converge, then say that $s = \infty$ — of course in that case the condition $R_{i,j}$ is certainly satisfied. (Note also that if any of the three are not in A_j, then condition $R_{i,j}$ is already satisfied. In fact, the proof will not make use of the fact that A_i is a proper subset of $\{0, 1\}^*$.) Furthermore, we are going to define $f(a_0, a_i) = a_i$ for all integers i and $f(a_{-1}, a_1) = a_0$ so we may assume that x_0 is the identity of A_i and that $f_i(x_{-1}, x_1) = x_0$. For each $k > 1$, let $x_{k+1} = f_j(x_k, x_1)$ and let $x_{-k-1} = f_j(x_{-k}, x_{-1})$ so that $x_n = nx_1$ for all integers n. Observe that the set $x_0, x_1, \ldots$ may be finite or infinite.

Now for ϕ_j to be an isomorphism from $\mathscr{A}$ onto A_j, then the order of a_1 in $C = C_{i,j}$ must equal the order of x_1 in A_j. We will ensure condition $R_{i,j}$ by forcing these two orders to be different.

We begin the definition of C and f by putting $a_m \in C$ for all m with $|m| < s$ and setting $f(a_m, a_n) = a_{m+n}$ for $|m| + |n| < s$. Since the elements of $\{a_m: |m| < s\}$ are all distinct, it follows that the order of a_1 is at least $2s - 1$. It follows that $\phi_j'(a_n) = x_n$ for all integers n with $|n| < s$ (or else condition $R_{i,j}$ is already satisfied.)

There are three cases.

Case 1. If the order of x_1 is $\leqslant 2s$ (that is, $\{x_m: |m| \leqslant s\}$ is not distinct), then we let $C_{i,j} = S_{i,j}$ and $f(a_m, a_n) = a_{m+n}$ for all $m, n \in \mathbb{Z}$. Thus the order of a_1 is infinite, so that ϕ_j' is not an isomorphism from $C_{i,j}$ into $\mathscr{A}_j$.

Case 2. If $s = \infty$, define $\mathscr{C}_{i,j}$ as in Case 1. Now ϕ_j' is not an isomorphism of $C_{i,j}$ into A_j since it is not even defined on all of $\{a_{-1}, a_0, a_1\}$.

Case 3. If the order of x_1 is $>2s$ (that is, $\{x_m: |m| \leqslant s\}$ is distinct), then we let $C_{i,j} = \{a_n: |n| < s\}$ and we let $f(a_m, a_n) = a_r$, where $|r| < s$ and $r = m + n$ modulo $2s$. Thus the order of a_1 equals $2s$, so that ϕ_j' is not an isomorphism from $C_{i,j}$ into $\mathscr{A}_j$.

In each case, it follows that $\mathscr{A}$ cannot be recursively isomorphic to any of the primitive recursive Abelian groups $(\mathscr{A}_i, f_i)$.

Let us check that the sequence $\mathscr{B}_n$ of Abelian groups is uniformly recursive. To show that the sets B_n are uniformly recursive, it suffices to show that the sets $C_{i,j}$ are uniformly recursive. Now given i, j and a natural number a in unary form, the test for possible membership of a in $C_{i,j}$ consists of the following. First compute m so that $a = \exp_3(m) = a_m$. Then perform m steps in the computations of $\phi_j'(a_k)$ for $k = -1, 0, 1$. If the computations do not yet converge, then $a \in C_{i,j}$. If they converge after a number of steps $s \leqslant |a|$, then compute the double sequence $x_n: |n| \leqslant s$ using the formulas $x_{n+1} = f_i(x_n, x_1)$ and $x_{-n-1} = f_i(x_{-n}, x_{-1})$. Finally, $a \in C$ if and only if the computed values x_n are distinct.

It remains to be seen that the functions $f_{i,j}$ are uniformly recursive. Given $a, b \in C_{i,j}$, we can compute the integers m, n so that $a = a_m$ and $b = a_n$ in the enumeration of $S_{i,j}$. We then perform $|m| + |n|$ steps in the computations of $\phi_j(a_k)$ for $k = -1, 0, 1$. If the computations fail to converge, then we set $f_{i,j}(a, b) = a_{m+n}$. If they converge, then we let s be the total length of the computations and define $f_{i,j}(a_m, a_n)$ as described above in Cases 1 and 3.

Now Lemma 3.5 completes the proof of part (i).

(ii) The proof of part (i) is again modified in several ways.

Let (A_0, f_0), (A_1, f_1), . . . be an effective list of all p-time Abelian groups and let $\phi_0, \phi_1, \ldots$ be a list of all one- to-one partial recursive functions as before.

As above, we will define, for each i, j, a set $D_{i,j}$ which is either S or a finite subset of S. We will then define a group structure $\mathscr{D}_{i,j}$ on each $D_{i,j}$ and let $\mathscr{A} = \sum_n \mathscr{G}_n$, where, for each n, $\mathscr{G}_n = \mathscr{D}_{(n)_1,(n)_2}$.

Define the double sequence $\{x_m: m \in \mathbb{Z}\}$ as in part (i).

The definition of $D_{i,j}$ proceeds as follows. For each m, perform the following series of computations in time bounded by a_m.

(1) Compute the values of x_k for $k \leq m$.

(2) Check whether these values are distinct.

If the computations are not completed in the prescribed time, or if the values are not distinct, then a_m and a_{-m} are put into $D = D_{i,j}$.

Let us assume, by way of eventual contradiction, that ϕ_j' is an isomorphism from $\mathscr{D}$ into $\mathscr{A}_j$. Then all values of x_k can be computed and the order of a_1 in $\mathscr{D}$ must equal the order of x_1 in $\mathscr{A}_j$.

We now show that the computations can be successfully completed for some m. As in the proof of part (ii) of Theorem 3.1, we let c be the amount of time required to compute $x_k = \phi_j(a_k)$ for $k = 0,\ 1,\ -1$ as well as to compute $f_i(y, z)$ for $\max(|y|, |z|) \leq 1$, and we let k be such that $f_i(y, z)$ can be computed in time $(\max(|y|, |z|)^k$ for any y and z with $\max(|y|, |z|) > 1$. Then to compute the double sequence $x_0, x_1, x_{-1}, x_2 = f_i(x_1, x_1), x_{-2} = f_i(x_{-1}, x_{-1}), \ldots, x_m = f_i(x_1, x_{m-1}), x_{-m} = f_i(x_{-1}, x_{-m+1})$ requires time at most

$$t(m) = c + b^k + (b^k)^k + \cdots$$
$$= c + b^k + b^{k^2} + \cdots + b^{k^m} \qquad (\text{where } b = |x_1| + |x_{-1}|).$$

As we saw in the proof of Theorem 3.1, $t(m)$ is dominated by $\exp_3(m) = a_m$.

Now let s be the least m such that the computations can be successfully completed.

It follows from the definition of D above that $a_m \in D$ for all m with $|m| < s$. We begin the definition of f by setting $f(a_m, a_n) = a_{m+n}$ for $|m| + |n| < s$. Since the elements of $\{a_m : |m| < s\}$ are all distinct, it follows that the order of a_1 is at least $2s - 1$. If ϕ_j is to be an isomorphism, it then follows that $\phi_j'(a_n) = x_n$ for all integers n with $|n| < s$.

There are three cases.

Case 1. If the order of x_1 is $\leq 2s$ (that is, $\{x_m : |m| \leq s\}$ is not distinct), then $D = S$ and we let $f(a_m, a_n) = a_{m+n}$ for all $m, n \in \mathbb{Z}$. Thus the order of a_1 is infinite, so that ϕ_j' is not an isomorphism from $\mathscr{D}_{i,j}$ into $\mathscr{A}_j$.

Case 2. If $s = \infty$, define $\mathscr{D}_{i,j}$ as in Case 1. Now ϕ_j' is not an isomorphism of $\mathscr{D}_{i,j}$ into $\mathscr{A}_j$ since it is not even defined on all of $\{a_{-1}, a_0, a_1\}$.

Case 3. If the order of x_1 is $> 2s$ (that is, $\{x_m : |m| \leq s\}$ is distinct), then $D = \{a_n : |n| < s\}$ and we let $f(a_m, a_n) = a_r$, where $|r| < s$ and $r = m + n$ modulo 2s. Thus the order of a_1 equals 2s, so that ϕ_j' is not an isomorphism from $\mathscr{D}_{i,j}$ into $\mathscr{A}_j$.

In each case, it follows that $\mathscr{A}$ cannot be recursively isomorphic to any of the p-time Abelian groups $(\mathscr{A}_i, f_i)$.

Let us check that the sequence $\mathscr{G}_n$ of Abelian groups is uniformly exponential-time. To show that the sets G_n are uniformly exponential-time, it suffices to show that the sets $D_{i,j}$ are uniformly exponential-time. Now given i, j and a natural

number a in unary form, the test for possible membership of a in $D_{i,j}$ consists of the following. First compute m so that $a = \exp_3(m) = a_m$. Then perform $|a|$ steps in the computation of the double sequence x_n: $|n| \leq m$. If the computations do not yet converge, then $a \in D$. If they converge after a number of steps $s \leq m$, then $m \in D$ if and only if the computed values x_n are distinct. Since the primary computations are bounded in advance by $|a|$, it is clear that D is actually polynomial time.

It remains to be seen that the functions $f_{i,j}$ are uniformly exponential-time. We make two key observations.

(1) The graph of $\exp_3$ is p-time.

(2) For any m and n, $\exp_3(m+n) < 2^{\exp_3(m)} + 2^{\exp_3(n)}$.

Given $a, b \in D_{i,j}$, we can compute the integers m, n so that $a = a_m$ and $b = a_n$ in the enumeration of S. This can be done in polynomial time by observation (1). Next we search up to $|m| + |n|$ for the least s for which the computation of the double sequence x_k: $|k| \leq s$ can be performed in time $|a_s|$. We then consider Cases 1, 2 and 3 above to determine the value of p such that $f(a, b) = a_p$. This can certainly be done in polynomial time. Finally we compute $f(a, b)$ as the least $c < \exp_3(m+n)$ such that $c = \exp_3(p)$. This can be done in exponential time by observation (2).

It follows from Lemma 3.5 that the direct sum $\mathscr{A}$ is exponential-time. This completes the proof of Theorem 3.6. □

4. A polynomial time model of arithmetic

In this section we construct a polynomial-time model of arithmetic which includes addition, multiplication and also the unary exponentiation function 2^x.

Theorem 4.1. *The structure* $(\mathbb{N}, S, +, \cdot, 2^x, <, 0)$ *for the language consisting of two unary function symbols, two binary function symbols, one binary relation symbol and one constant symbol, is recursively isomorphic to a polynomial-time structure.*

The remainder of the section is devoted to the proof of this theorem.

Let the language $\mathscr{L} = \{0, A, E\}$, where 0 is a constant symbol (representing itself), A is a binary function symbol (representing addition) and E is a unary function symbol (representing the function 2^x). Letting exponentiation take precedence over addition, we can omit parentheses and write Ex for $E(x)$ and Axy for $A(x, y)$. We can define the natural number $n(\sigma)$ represented by a term σ by the following recursion scheme:

$$n(0) = 0,$$
$$n(E\sigma) = 2^{n(\sigma)} \quad \text{for any term } \sigma,$$
$$n(A\sigma\tau) = n(\sigma) + n(\tau) \quad \text{for any terms } \sigma \text{ and } \tau.$$

It is clear that every natural number n is represented by many terms. For example, the number n can be written in 'unary' form as the sum of n 1s. More importantly, the standard binary representation for a natural number $n>0$ has the form $n=2^{x_1}+\cdots+2^{x_k}$, where $x_1>x_2>\cdots>x_k$; the numbers $x_1,\ldots,x_k$ can be represented in 'unary' form or, by recursion, in binary form. For example, the number 5 can be represented by $AEAE0E0E0$ and also $AEEE0E0$. We want to define by recursion an efficient representation $\sigma(n)$ of a natural number n by a term in the languge $\mathscr{L}$. Note first that any number $n>0$ may be written uniquely in the form $n=2^k+m$ where $m<2^k$. This will imply that the function σ defined below is well-defined for all natural numbers n.

$$\sigma(0)=0,$$

$$\sigma(2^k)=E\sigma(k) \quad \text{for any } k,$$

$$\sigma(2^k+m)=AE\sigma(k)\sigma(m) \quad \text{for } 0<m<2^k.$$

Let T be the set of terms of $\mathscr{L}$. It is clear that $n(\sigma)$ is a recursive function from T into $\mathbb{N}$. This function of course cannot be polynomial- or even exponential-time, since it maps the string $E^n2=E^n(EE0)$ (which has length $n+3$) to the number $2^{2^{\cdots}}$.

We can now define the polynomial-time structure $\mathscr{M}=(M, S^M, +^M, \cdot^M, E^M, <^M, 0)$ which is isomorphic to $(\mathbb{N}, S, +, \cdot, 2^x, <, 0)$. The domain M is $\{\sigma(n): n\in\mathbb{N}\}$, the unary operations are defined by $S^M(\sigma(n))=\sigma(n+1)$ and $E^M(\sigma(n))=\sigma(2^n)$; the binary operations are defined by $\sigma(m)+^M\sigma(n)=\sigma(m+n)$ and $\sigma(m)\cdot^M\sigma(n)=\sigma(m\cdot n)$, the relation $<^M$ is defined by $\sigma(m)<^M\sigma(n)\Leftrightarrow m<n$ and the constant $0^M=0$.

It is clear that σ is a recursive isomorphism from $\mathbb{N}$ onto M. It remains to be seen that the domain M, the relation $<^M$, and the functions S^M, $+^M$, $\cdot^M$ and E^M are polynomial-time.

Let us first observe that the set T is polynomial-time, by the following standard algorithm. Given an input word σ, read the word from right to left and keep track of a counter C as follows. Initially $C=0$. When a constant is read, increment C by 1. When an E is read, leave C unchanged. When an A is read, decrement C by 1. Now $\sigma\in T$ as long as the counter ends up at 1 and is never 0 after any letters are read. It is clear that this algorithm requires only $c\cdot|(\sigma)|$ time, where c is some small constant. This demonstrates the following lemma.

Lemma 4.2. *T is linear-time.*

This allows us to define our structure relative to T. We need one more property of T, which is standard for a set of terms in Polish notation such as this. (See Enderton [1, pp. 97–100].)

Lemma 4.3 (Unique Readability). (i) *No proper initial segment of a term $\tau \in T$ is a term.*

(ii) *Every term $\sigma \in T$ is either 0, $E\tau$ for some $\tau \in T$, or $A\tau\rho$ for some unique $\tau, \rho \in T$.*

Because of Lemma 4.3(i), we can recover τ and ρ from a term $A\tau\rho$ in linear time in $|A\tau\rho|$. That is, given a term $\sigma = A\tau\rho$, first strip off the initial A. Now read the word $\tau\rho = \gamma_0\gamma_1 \cdots \gamma_n$ from left to right and keep track of a counter D as follows. Initially $D = 0$; when a constant 0 is read, increment D by 1; when E is read, leave D unchanged; when A is read, decrement D by 1. Let k be the least i such that $\gamma_i = 0$ and $D = 1$ after we read γ_i and $D \leq 0$ after we read γ_j for each $j < i$. It is then easy to see that $\gamma_0 \cdots \gamma_k$ is the first initial segment of $\tau\rho$ which is a term. But then by Lemma 4.3(i), we must have $\gamma_0, \ldots, \gamma_k = \tau$. Finally $\gamma_{k+1} \cdots \gamma_n = \rho$.

It follows that we may define functions by recursion on T and prove propositions by induction on T.

Lemma 4.4. *The function σ maps $\mathbb{N}$ one-to-one into T.*

Proof. The proof that σ is one-to-one is by induction. Let $P(n)$ be the statement that for all n_1 and n_2 with $\min\{n_1, n_2\} \leq n$, $\sigma(n_1) = \sigma(n_2)$ implies $n_1 = n_2$. $P(0)$ is clear. Suppose therefore that $P(n)$ holds and that $\sigma(n_1) = \sigma(n_2)$ for some n_1 and n_2 with $\min\{n_1, n_2\} = n_1 = n + 1$. There are two cases.

(1) $n_1 = 2^{k_1}$ for some $k_1 \leq n$. Then $\sigma(n_1) = E\sigma(k_1) = \sigma(n_2)$. Since $\sigma(n_2)$ starts with E, it is clear that $n_2 = 2^{k_2}$ for some k_2 such that $\sigma(k_1) = \sigma(k_2)$. It follows by induction that $k_1 = k_2$, so that $n_1 = n_2$.

(2) $n_1 = 2^{k_1} + m_1$ for some $k_1, m_1 \leq n$ with $0 < m_2 < 2^{k_1}$. Then $\sigma(n_1) = AE\sigma(k_1)\sigma(m_1) = \sigma(n_2)$. It follows by unique readability that $n_2 = 2^{k_2} + m_2$ for some k_2, m_2 with $0 < m_2 < 2^{k_2}$ such that $\sigma(m_1) = \sigma(m_2)$ and $\sigma(k_1) = \sigma(k_2)$. It follows by induction that $k_1 = k_2$ and $m_1 = m_2$, so that $n_1 = n_2$. □

Definition 4.1. For any words σ and τ in M and $\rho \in T$,

(a) $T_M(\rho)$ is the time required to check whether $\rho \in M$,

(b) $T_S(\sigma)$ is the time required to compute $S\sigma$,

(c) $T_A(\sigma, \tau)$ is the time required to compute $\sigma +^M \tau$,

(d) $T_P(\sigma, \tau)$ is the time required to compute $\sigma \cdot^M \tau$,

(e) $T_L(\sigma, \tau)$ is the time required to compare σ and τ and give the resulting answer $\sigma <^M \tau$, $\sigma = \tau$ or $\tau <^M \sigma$.

In cases (c), (d) and (e) we input σ on one tape, input τ on another tape, and write the answer on a third tape. These tapes are assumed to have heads which can move independently.

We now begin the process of showing that the structure $\mathcal{M}$ is p-time by showing that the set M and the binary relation $<^M$ are p-time.

Lemma 4.5. *There is a constant c such that, for any* ρ_1 *and* ρ_2 *in* M,

$$T_L(\rho_1, \rho_2) \leqslant c(|\rho_1| + |\rho_2|)^2$$

Proof. First observe that $<^M$ may be defined recursively by the following four clauses:

(1) $0 <^M \sigma$, for all $\sigma \neq 0$.

(2) $E\sigma <^M E\tau \Leftrightarrow \sigma <^M \tau$.

(3) $E\sigma <^M AE\tau\rho \Leftrightarrow \sigma \leqslant^M \tau$.

(4) $AE\sigma\pi <^M AE\tau\rho \Leftrightarrow [\sigma <^M \tau$ or $(\sigma = \tau$ and $\pi <^M \rho)]$.

These can be demonstrated as follows:

Proof of (1). Since $\sigma \neq 0$ and $\sigma \in M$, we have $\sigma = \sigma(n)$ for some n, where $n = 2^k + m$ for some k and m. Clearly $0 < n$, so that $0 <^M \sigma$.

Proof of (2). Since σ and τ are in M, there exist m and n such that $\sigma = \sigma(m)$ and $\tau = \sigma(n)$ and therefore $\sigma(2^m) = E\sigma$ and $\sigma(2^m) = E\tau$. Now

$$E\sigma <^M E\tau \Leftrightarrow 2^m < 2^n \Leftrightarrow m < n \Leftrightarrow \sigma <^M \tau.$$

Proof of (3). Let $\sigma = \sigma(n)$, $\tau = \sigma(k)$ and $\rho = \sigma(m)$, so that $E\sigma = \sigma(2^n)$ and $AE\tau\rho = \sigma(2^k + m)$. Then

$$\sigma \leqslant^M \tau \Leftrightarrow n \leqslant k \Leftrightarrow 2^n < 2^k + m \Leftrightarrow E\sigma <^M AE\tau\rho.$$

Proof of (4). Let $\sigma = \sigma(j)$, $\pi = \sigma(n)$, $\tau = \sigma(k)$ and $\rho = \sigma(m)$, so that $AE\sigma\pi = \sigma(2^j + n)$ and $AE\tau\rho = \sigma(2^k + m)$. Then

$$\begin{aligned} AE\sigma\pi <^M AE\tau\rho &\Leftrightarrow 2^j + n < 2^k + m \Leftrightarrow [j < k \vee (j = k \wedge n < m)] \\ &\Leftrightarrow [\sigma <^M \tau \vee (\sigma = \tau \wedge \pi <^M \rho)]. \end{aligned}$$

From this recursive definition of $<^M$ we can now prove the lemma by induction on ρ_1 and ρ_2 in 4 cases. Each of the four cases provides a lower bound for the desired constant c.

(1) To compare σ with 0, we just need to read the first symbol of σ to decide whether $0 = \sigma$; if it doesn't, then $0 <^M \sigma$. Therefore $T_L(0, \sigma)$, $T_L(\sigma, 0) \leqslant c$ for some constant c.

(2) To compare $E\sigma$ and $E\tau$, we just mark the two Es and then apply the procedure to σ and τ. It follows that

$$T_L(E\sigma, E\tau),\ T_L(E\tau, E\sigma) \leqslant c + T_L(\sigma, \tau).$$

Now we have by induction that $T_L(\sigma, \tau) \leqslant c(|\sigma| + |\tau|)^2$. Therefore

$$\begin{aligned} T_L(E\sigma, E\tau) &\leqslant c + T_L(\sigma, \tau) \\ &\leqslant c + c(|\sigma| + |\tau|)^2 \\ &\leqslant c(|E\sigma| + |E\tau|)^2. \end{aligned}$$

(3) To compare $E\sigma$ and $AE\tau\rho$, we have to find σ and τ and copy them onto two auxiliary tapes, then apply the procedure to compare σ and τ. We just look at one symbol of ρ to check that ρ is not missing. It follows that

$$T_L(E\sigma, AE\tau\rho),\ T_L(AE\tau\rho, E\sigma) \leqslant T_L(\sigma, \tau) + c(|\sigma| + |\tau|).$$

Again we have by induction that $T_L(\sigma, \tau) \leqslant (|\sigma| + |\tau|)^2$. Let $s = |\sigma|$ and $t = |\tau|$. Therefore

$$\begin{aligned} T_L(E\sigma, AE\tau\rho) &\leqslant T_L(\sigma, \tau) + c(|\sigma| + |\tau|) \\ &\leqslant c(s+t)^2 + c(s+t) \\ &= c(s^2 + t^2 + 2st + s + t) \\ &\leqslant c(s+t+3)^2 \\ &\leqslant c(|E\sigma| + |AE\tau\rho|)^2. \end{aligned}$$

(4) To compare $AE\sigma\pi$ with $AE\tau\rho$, we have to find σ, τ, ρ and π, then apply the procedures to compare σ with τ and (possibly) to compare ρ with π. It follows that

$$T_L(AE\sigma\pi, AE\tau\rho) \leqslant T_L(\sigma, \tau) + T_L(\pi, \rho) + c(|\sigma| + |\pi| + |\tau| + |\rho|).$$

Finally, we have by induction that $T_L(\sigma, \tau) \leqslant (|\sigma| + |\tau|)^2$ and also that $T_L(\pi, \rho) \leqslant (|\pi| + |\rho|)^2$. Let $s = |\sigma|$, $t = |\tau|$, $p = |\pi|$ and $r = |\rho|$. Then

$$\begin{aligned} T_L(AE\sigma\pi, AE\tau\rho) &\leqslant T_L(\sigma, \tau) + T_L(\pi, \rho) + c(|\sigma| + |\pi| + |\tau| + |\rho|) \\ &\leqslant c(s+t)^2 + c(p+r)^2 + c(s+t+p+r) \\ &\leqslant c(s+t+p+r+4)^2 \\ &= c(|AE\sigma\pi| + |AE\tau\rho|)^2. \end{aligned}$$

This completes the proof of Lemma 4.5. □

Lemma 4.6. *There is a constant c such that, for any ρ in T,*

$$T_M(\rho) \leqslant c\,|\rho|^3.$$

Proof. The proof is again by induction on the length of σ and depends on Lemma 4.5.

First observe that M may be defined recursively by the following three clauses:

(1) $0 \in M$.

(2) $E\sigma \in M \Leftrightarrow \sigma \in M$.

(3) $AE\sigma\tau \in M \Leftrightarrow \sigma \in M \wedge \tau \in M \wedge \tau \neq 0 \wedge \tau <^M E\sigma$.

These can be demonstrated as follows.

Proof of (1). $0 \in M$ because $n(0) = 0$.

Proof of (2). If $\sigma \in M$, then $\sigma = \sigma(n)$ for some n. It follows that $E\sigma = \sigma(2^n)$, so that $E\sigma \in M$. If $E\sigma \in M$, then $E\sigma = \sigma(2^k)$ for some k with $\sigma = \sigma(k)$, so that $\sigma \in M$.

Proof of (3). If $\sigma, \tau \in M$ and $\tau \neq 0$ and $\tau <^M E\sigma$, then $\sigma = \sigma(k)$ for some k and $\tau = \sigma(m)$ for some m and k such that, by Lemma 4.5, $0 < m < 2^k$. It now follows that $AE\sigma\tau = \sigma(2^k + m)$, so that $AE\sigma\tau \in M$. If $AE\sigma\tau \in M$, then $AE\sigma\tau = \sigma(2^k + m)$ for some m and k with $0 < m < 2^k$, $\sigma = \sigma(k)$ and $\tau = \sigma(m)$. It follows that σ and τ are in M, that $\tau \neq 0$ and $\tau <^M E\sigma$.

From this recursive definition of M we can now prove the lemma by induction on ρ in three cases. Each of the three cases provides a lower bound for the desired constant c.

(1) We are given that $0 \in M$, so that

$$T_M(0) \leqslant c.$$

(2) To test whether $E\sigma \in M$, we observe the E at the beginning, mark the square on the tape and then apply the procedure to the remainder of the tape. It follows that

$$T_M(E\sigma) \leqslant c + T_M(\sigma).$$

Now we have by induction that $T_M(\sigma) \leqslant c(|\sigma|)^3$. Therefore

$$T_M(E\sigma) \leqslant c + T_M(\sigma) \leqslant c + c\,|\sigma|^3 \leqslant c(1 + |\sigma|)^3 = |E\sigma|^3.$$

(3) To test whether $AE\sigma\tau \in M$, we have to find σ and τ and copy them onto two auxiliary tapes, then apply the procedures to test whether σ and τ are in M and to compare σ and τ. It follows that

$$T_M(AE\sigma\tau) \leqslant T_M(\sigma) + T_M(\tau) + T_L(E\sigma, \tau) + c(|\sigma| + |\tau|).$$

Now we have by induction that $T_M(\sigma) \leqslant c\,|\sigma|^3$ and $T_M(\tau) \leqslant c\,|\tau|^3$ and we know from Lemma 4.5 that $T_L(E\sigma, \tau) \leqslant c(|E\sigma| + |\tau|)^2$. Therefore

$$\begin{aligned} T_M(AE\sigma\tau) &\leqslant T_M(\sigma) + T_M(E\tau) + T_L(E\sigma, \tau) + c(|\sigma| + |\tau|) \\ &\leqslant c\,|\sigma|^3 + c\,|\tau|^3 + c(|E\sigma| + |\tau|)^2 + c(|\sigma| + |\tau|) \\ &\leqslant c(|\sigma|^3 + |\tau|^3 + (|\sigma| + |\tau| + 1|)^2 + |\sigma| + |\tau|) \\ &\leqslant c(|\sigma| + |\tau| + 2)^3 = c\,|AE\sigma\tau|^3. \end{aligned}$$

This completes the proof of Lemma 4.6. □

Next we will consider the successor function.

Lemma 4.7. (i) *For any* $\rho \in M$, $|S^M(\rho)| \leqslant |\rho| + 3$.

(ii) *There is a constant* k *such that, for any* $\rho \in M$, $T_S(\rho) \leqslant k\,|\rho|^3$.

Proof. We begin with a recursive procedure for computing the successor $S^M(\rho)$ of a term $\rho \in M$. There are three cases.

(0) $S^M(0) = E0$; $S^M(E0) = EE0$.

(1) For $\sigma \neq 0$, $E0$, $S^M(E\sigma) = AE\sigma E0$.

(2) For any term $AE\sigma\tau \in M$, $S^M(AE\sigma\tau)$ is determined as follows: First compute $S^M(\tau)$ and then compare $S^M(\tau)$ with $E\sigma$. There are two sub-cases.

(a) If $S^M(\tau) <^M E\sigma$, then $S^M(AE\sigma\tau) = AE\sigma S^M(\tau)$.

(b) If $S^M(\tau) = E\sigma$, then compute $S^M(\sigma)$. Now $S^M(AE\sigma\tau) = ES^M(\sigma)$.

This procedure can be verified by expressing each term in M as the image $\sigma(n)$ of some natural number. We will give the proof of (2) and leave the easier cases to the reader.

(2) Let $\sigma = \sigma(k)$ and $\tau = \sigma(m)$, so that $0 < m < 2^k$ and $AE\sigma\tau = \sigma(n)$, where $n = 2^k + m$. Then of course $S^M(AE\sigma\tau) = \sigma(n+1)$. Now $S^M(\tau) = m + 1 \leqslant 2^k$. The two cases are (a) when $m + 1 < 2^k$ and (b) when $m + 1 = 2^k$.

In case (a), we have

$$S^M(AE\sigma\tau) = \sigma(2^k + m + 1) = AE\sigma(k)\sigma(m+1) = AE\sigma S^M(\tau).$$

In case (b), we have $n + 1 = 2^k + 2^k = 2^{k+1}$ and $S^M(\sigma) = \sigma(k+1)$, so that

$$S^M(AE\sigma\tau) = \sigma(2^{k+1}) = ES^M(\sigma).$$

It is now easy to prove (i) by induction on ρ. There are three cases as described above.

(0) If $\rho = 0$ or $\rho = E0$, then $|S^M(\rho)| = |\rho| + 1$.

(1) If $\rho = E\sigma$ and $\sigma \neq 0$, $E0$, then $|S^M(\rho)| = |\rho| + 3$.

(2) If $\rho = AE\sigma\tau$, then there are two cases.

(a) $|S^M(\rho)| = |\sigma| + |S^M(\tau)| + 2$, which is $\leqslant |\sigma| + |\tau| + 5$ by induction.

(b) $|S^M(\rho)| = |S^M(\sigma)| + 1$, which is $\leqslant |\sigma| + 4$ by induction.

In both cases, the length is $\leqslant |\sigma| + |\tau| + 5 = |\rho| + 3$, as desired.

Now we turn to the proof of (ii), again proved by recursion on ρ in three cases. Each case provides a lower bound for the desired constant k.

(0) $S^M(0) = E0$ can be computed in some fixed finite time k.

(1) To compute $S^M(E\sigma)$, we just check to see whether $\sigma = 0$; if it is, we write $E0$ and if not, we write $AE\sigma E0$. It follows that

$$T_S(E\sigma) \leqslant k\,|\sigma|\,.$$

(2) To compute $S^M(AE\sigma\tau)$, we have to compute $S^M(\tau)$ and, possibly, $S^M(\sigma)$ and also compare $S^M(\tau)$ with $E\sigma$. Since, by part (a), $|S^M(\tau)| \leqslant |\tau| + 3$, the comparison can be done, by Lemma 4.5, in time $\leqslant c(|\sigma| + |\tau| + 4)^2$, which is $\leqslant 4c(|\sigma| + |\tau|)^2$ since $\sigma, \tau \neq 0$. To perform these computations we also need to make copies of σ and τ and write the final answer. It follows that for some fixed k,

$$T_S(AE\sigma\tau) \leqslant T_S(\tau) + T_S(\sigma) + k(|\sigma| + |\tau|)^2.$$

Now by induction, we know that $T_S(\tau) \leqslant k\,|\tau|^3$ and that $T_S(\sigma) \leqslant k\,|\sigma|^3$. Therefore

$$\begin{aligned} T_S(\rho) &\leqslant k(|\sigma|^3 + |\tau|^3 + (|\sigma| + |\tau|)^2) \\ &\leqslant k(|\sigma| + |\tau|)^3 = k\,|\rho|^3. \end{aligned}$$

This completes the proof of Lemma 4.7. □

Next we will consider the addition function.

We will need the following inequality.

Lemma 4.8. *For any natural numbers n, x and y with $x < y$ and any real number $r > 0$,*

$$x^{r+1} + y^r \leq y^{r+1}.$$

Proof. Since $x < y$ and $r > 0$, we have $x^r < y^r$. It follows that $x^{r+1} = x^r x < y^r x$, so that

$$x^{r+1} + y^r < y^r(x+1) \leq y^r y = y^{r+1}. \qquad \square$$

Lemma 4.9. (A) *For any terms ρ_1 and ρ_2 in M, $|\rho_1 +^M \rho_2| \leq |\rho_1| + |\rho_2| + 1$.*

(B) *There is a constant k so that for any $\rho_1, \rho_2 \in M$, $T_A(\rho_1, \rho_2) \leq k(|\rho_1| + |\rho_2|)^4$.*

Proof. The proof is by induction on ρ_1 and ρ_2. We begin with the procedure for computing the sum $\sigma +^M \tau$ is also an implicit definition of $\tau +^M \sigma$.

(0) For any term σ, $\sigma +^M 0 = \sigma$.

(1) For any two terms σ, τ the sum of $E\sigma$ and $E\tau$ is determined as follows. First compare the terms σ and τ. There are three cases.

(a) If $\sigma < \tau$, then $E\sigma +^M E\tau = AE\tau E\sigma$;

(b) If $\tau < \sigma$, then $E\sigma +^M E\tau = AE\sigma E\tau$;

(c) If $\sigma = \tau$, then compute $S^M(\sigma)$; now $E\sigma +^M E\tau = ES^M(\sigma)$.

(2) For any terms σ and $\eta = AE\tau\rho$, the sum $\xi = E\sigma +^M \eta$ is determined as follows. First compare σ and τ. There are three cases.

(a) If $\tau < \sigma$, then $\xi = AE\sigma\eta$;

(b) If $\tau = \sigma$, then compute $S^M(\sigma)$; now $\xi = AES^M(\sigma)\rho$.

(c) If $\sigma < \tau$, then compute $E\sigma +^M \rho$ and compare the result with $E\tau$; there are three subcases:

(i) If $E\sigma +^M \rho < E\tau$, then $\xi = AE\tau(E\sigma +^M \rho)$;

(ii) If $E\sigma +^M \rho = E\tau$, then compute $S^M(\tau)$; now $\xi = ES^M(\tau)$.

(iii) If $E\sigma +^M \rho > E\tau$, then $E\sigma +^M \rho = AE\tau\pi$ for some π; compute $S^M(\tau)$ again; now $\xi = AES^M(\tau)\pi$.

(3) For any two terms $\rho_1 = AE\sigma_1\tau_1$ and $\rho_2 = AE\sigma_2\tau_2$, the sum $\xi = \rho_1 +^M \rho_2$ is determined as follows. First compare σ_1 and σ_2. There are three cases.

(a) If $\sigma_1 = \sigma_2$, then compute $S^M(\sigma_1)$ and $\tau_1 +^M \tau_2$; then $\xi = AES^M(\sigma_1)(\tau_1 +^M \tau_2)$.

(b) If $\sigma_1 < \sigma_2$, then compute $\rho = \rho_1 +^M \tau_2$ and compare ρ with $E\sigma_2$. There are three subcases.

(i) If $\rho < E\sigma_2$, then $\xi = AE\sigma_2\rho$;

(ii) If $\rho = E\sigma_2$, then compute $S^M(\sigma_2)$; now $\xi = ES^M(\sigma_2)$.

(iii) If $\rho > E\sigma_2$, then $\rho = AE\sigma_2\pi$ for some π; compute $S^M(\sigma_2)$ again; now $\xi = AES^M(\sigma_2)\pi$.

(c) If $\sigma_2 <^M \sigma_1$, then proceed as in (b) by symmetry.

As in Lemma 4.7, this procedure can be verified by expressing each term in M as the image $\sigma(n)$ of some natural number. We will give the proof of (2) and leave the other cases to the reader.

Let $\sigma = \sigma(i)$, $\tau = \sigma(k)$ and $\rho = \sigma(m)$, so that $0 < m < 2^k$. Then $E\sigma +^M \eta = \sigma(2^i + 2^k + m)$ and the cases are as follows.

(a) If $k < i$, then $2^k + m < 2^k + 2^k = 2^{k+1} \leq 2^i$, so that

$$E\sigma +^M \eta = \sigma(2^i + 2^k + m) = AE\sigma(i)\sigma(2^k + m) = AE\sigma\eta.$$

(b) If $k = i$, then $2^i + 2^k + m = 2^{k+1} + m$ and $m < 2^{i+1}$, so that

$$E\sigma +^M \eta = \sigma(2^{i+1} + m) = AES^M(\sigma)\rho.$$

(c) If $i < k$, then we have to compare $2^i + m$ with 2^k, with three possible cases.

(i) If $2^i + m < 2^k$, then we have

$$E\sigma +^M \eta = \sigma(2^k + 2^i + m) = AE\tau(E\sigma +^M \rho).$$

(ii) If $2^i + m = 2^k$, then $2^i + 2^k + m = 2^{k+1}$, so that

$$E\sigma +^M \eta = \sigma(2^{k+1}) = ES^M(\tau).$$

(iii) If $2^k < 2^i + m$, then since $i < k$ and $m < 2^k$, we have $2^k < 2^i + m < 2^{k+1}$, so that $2^i + m = 2^k + j$ for some j and $2^i + 2^k + m = 2^{k+1} + j$. Letting $\pi = \sigma(j)$, we have

$$E\sigma +^M \eta = \sigma(2^{k+1} + j) = AES^M(\tau)\pi.$$

We can now prove part (A).

The cases are as described above.

(0) $|\sigma +^M 0| = |\sigma| \leq |\sigma| + |0| + 1$.

(1) (a), (b) $|E\sigma +^M E\tau| = |E\sigma| + |E\tau| + 1$.

(c) $|E\sigma +^M E\tau| = |S^M(\sigma)| + 1$

$\leq |\sigma| + 4$ (by Lemma 4.7)

$\leq |E\sigma| + |E\tau| + 1$.

(2) Let $\theta = E\sigma$ and $\eta = AE\tau\rho$.

(a) $|\theta +^M \eta| = |\theta| + |\eta| + 1$.

(b) $|\theta +^M \eta| = |S^M(\sigma)| + |\rho| + 2$

$\leq |\sigma| + |\rho| + 5$ (by Lemma 4.7)

$\leq |\sigma| + |\tau| + |\rho| + 4 = |\theta| + |\eta|$.

(c)

(i) $|\theta +^M \eta| = |E\sigma +^M \rho| + |\tau| + 2$

$\leq |\sigma| + |\rho| + |\tau| + 4$ (by induction)

$= |\theta| + |\eta| + 1$.

(ii) $|\theta +^M \eta| = |S^M(\tau)| + 1$

$\leq |\tau| + 4$ (by Lemma 4.7)

$\leq |\theta| + |\eta| + 1.$

(iii) By Lemma 4.7, we have $|S^M(\tau)| \leq |\tau| + 3$ and by induction we have

$$|\tau| + |\pi| + 2 = |AE\tau\pi| = |E\sigma +^M \rho| \leq |\sigma| + |\rho| + 2.$$

It follows that $|\pi| \leq |\sigma| + |\rho| - |\tau|$, so that

$$\begin{aligned} |\theta +^M \eta| &= |S^M(\tau)| + |\pi| + 2 \\ &\leq |\tau| + 3 + |\sigma| + |\rho| - |\tau| + 2 \\ &= |\sigma| + |\rho| + 5 \leq |\theta| + |\eta| + 1. \end{aligned}$$

(3) Let $\rho_1 = AE\sigma_1\tau_1$ and $\rho_2 = AE\sigma_2\tau_2$

(a) By Lemma 4.7, we have $|S^M(\sigma_1)| \leq |\sigma_1| + 3$ and by induction we have

$$|\tau_1 +^M \tau_2| \leq |\tau_1| + |\tau_2| + 1.$$

It follows that

$$\begin{aligned} |\rho_1 +^M \rho_2| &\leq |\sigma_1| + |\tau_1| + |\tau_2| + 6 \\ &\leq |\sigma_1| + |\tau_1| + |\sigma_2| + |\tau_2| + 5 \\ &= |\rho_1| + |\rho_2| + 1. \end{aligned}$$

(b) By Lemma 4.7, we have $|S^M(\sigma_2) \leq |\sigma_2| + 3$ and by induction we have

$$|\rho_1 +^M \tau_2| \leq |\rho_1| + |\tau_2| + 1.$$

We now have three cases.

(i) $|\rho_1 +^M \rho_2| \leq |\sigma_2| + |\rho_1| + |\tau_2| + 3.$

(ii) $|\rho_1 +^M \rho_2| \leq |\sigma_2| + 4.$

(iii) $|\sigma_2| + |\pi| + 2 = |\rho| \leq |\rho_1| + |\tau_2| + 1$, so that $|\pi| \leq |\rho_1| + |\tau_2| - |\sigma_2| - 1$ and

$$\begin{aligned} |\rho_1 +^M \rho_2| &\leq |\sigma_2| + |\pi| + 5 \leq |\rho_1| + |\tau_2| + 4 \\ &\leq |\rho_1| + |\sigma_2| + |\tau_2| + 3 = |\rho_1| + |\rho_2| + 1. \end{aligned}$$

(c) Similar to (b).

This completes the proof of part (A).

We can now give the proof of part (B). As in part (A), this is proved by induction on ρ_1 and ρ_2 and the cases are based on the recursive procedure described above. As in Lemma 4.7, each case provides a lower bound for the desired constant k.

(0) Since $\sigma +^M 0 = \sigma$, it is clear that for some k,

$$T_A(0, \sigma) \leq k\,|\sigma|\,.$$

(1) To compute $E\sigma +^M E\tau$, we have to compare σ and τ, which can be done in quadratic time by Lemma 4.5 and then possibly compute $S^M(\sigma)$, which can be done in cubic time by Lemma 4.7. It follows that for some k,

$$T_A(E\sigma, E\tau) \leq k(|\sigma| + |\tau|)^3.$$

(2) To compute $E\sigma +^M AE\tau\rho$, we have to

(a) compare σ with τ, which takes quadratic time by Lemma 4.5;

(b) possibly compute $S^M(\sigma)$ or $S^M(\tau)$, which can be done in cubic time by Lemma 4.7;

(c) possibly compute $E\sigma +^M \rho$ and then compare the result with $E\tau$. By part (A), $|E\sigma +^M \rho| \leq |\sigma| + |\rho| + |2|$, so this comparison can be done in quadratic time by Lemma 4.5.

It follows that for some c,

$$T_A(E\sigma, AE\tau\rho) \leq T_A(E\sigma, \rho) + c(|E\sigma| + |AE\tau\rho|)^3.$$

Now by induction $T_A(E\sigma, \rho) \leq k(|\sigma| + |\tau|)^4$, so that, taking $k \geq c$, we have

$$\begin{aligned} T_A(E\sigma, AE\tau\rho) &\leq k[(|\sigma| + |\rho| + 1)^4 + (|E\sigma| + |AE\tau\rho|)^3] \\ &\leq k(|E\sigma| + |AE\tau\rho|)^4 \qquad \text{(by Lemma 4.8).} \end{aligned}$$

(3) Let $\rho_1 = AE\sigma_1\tau_1$, $\rho_2 = AE\sigma_2\tau_2$. To compute $\rho_1 +^M \rho_2$, we have to

(a) compare σ_1 with σ_2, which takes quadratic time by Lemma 4.5;

(b) possibly compute $S^M(\sigma_1)$ or $S^M(\sigma_2)$, which takes cubic time by Lemma 4.7;

(c) Compute either $\tau_1 +^M \tau_2$, $\tau_1 +^M \rho_2$, or $\rho_1 +^M \tau_2$ and compare the result, in the second case, with $E\sigma_2$ or, in the third case, with $E\sigma_1$. Since the result is bounded in length by $|\rho_1| + |\rho_2|$ by part (A), the comparison can be done in quadratic time, by Lemma 4.5.

It follows that

$$T_A(\rho_1, \rho_2) \leq c(|\rho_1| + |\rho_2|)^3 + t,$$

where t is one of $T_A(\tau_1, \tau_2)$, $T_A(\rho_1, \tau_2)$ or $T_A(\tau_1, \rho_2)$.

Now by induction, we know that $t \leq k(|\rho_1| + |\rho_2| - 2)^4$. Therefore, again taking $k \geq c$, we have

$$\begin{aligned} T_A(\rho_1, \rho_2) &\leq c(|\rho_1| + |\rho_2|)^3 + k(|\rho_1| + |\rho_2| - 2)^4 \\ &\leq k(|\rho_1| + |\rho_2|)^4 \qquad \text{(by Lemma 4.8).} \end{aligned}$$

This completes the proof of Lemma 4.9. □

Finally, we turn to multiplication.

Lemma 4.10. (a) *For any terms ρ_1 and ρ_2 in M, $|\rho_1 \cdot^M \rho_2| \leq |\rho_1|\,|\rho_2|$.*

(b) *There is a constant c such that for any ρ_1 and ρ_2, $T_P(\rho_1, \rho_2) \leq c\,|\rho_1|^5\,|\rho_2|^5$.*

Proof. The procedure for computing the product of two terms ρ_1 and ρ_2 is the following.

(0) For any term σ,

$$\sigma \cdot^M 0 = 0 \cdot^M \sigma = 0.$$

(1) For any two terms σ and τ, the product

$$E\sigma \cdot^M E\tau = E(\sigma +^M \tau).$$

(2) For any terms σ, τ and ρ, the product

$$AE\sigma\tau \cdot^M \rho = \rho \cdot^M AE\sigma\tau = (E\sigma \cdot^M \rho) +^M (\tau \cdot^M \rho).$$

This procedure is easily verified by expressing each term in the form $\sigma(n)$ for some natural number n. The details are left to the reader.

We can now give the proof of part (a). The cases are as described above in the recursive procedure for computing the product of two terms.

(0) $|\sigma \cdot^M 0| = 1 \leqslant |\sigma|\,|0|$.

(1) $$\begin{aligned} |E\sigma \cdot^M E\tau| &= |\sigma +^M \tau| + 1 \\ &\leqslant |\sigma| + |\tau| + 2 \qquad \text{(by Lemma 4.9)} \\ &= |E\sigma| + |E\tau| \leqslant |E\sigma|\,|E\tau|. \end{aligned}$$

(2) $$\begin{aligned} |AE\sigma\tau \cdot^M \rho| &\leqslant |E\sigma \cdot^M \rho| + |\tau \cdot^M \rho| + 1 \qquad \text{(by Lemma 4.9)} \\ &\leqslant (|\sigma| + 1)\,|\rho| + |\tau|\,|\rho| + 1 \qquad \text{(by induction)} \\ &= (|\sigma| + |\tau| + 1)\,|\rho| + 1 \\ &\leqslant (|\sigma| + |\tau| + 2)\,|\rho| = |AE\sigma\tau|\,|\rho|. \end{aligned}$$

We can now give the proof of (b). As in Lemma 4.9, the proof is by induction on ρ_1 and ρ_2 and the cases are based on the recursive procedure described above, with each case providing a lower bound for the desired constant c.

(0) Since $0 \cdot^M \sigma = 0$, it is clear that, for some c,

$$T_P(0, \sigma) \leqslant c.$$

(1) To compute $E\sigma \cdot^M E\tau$, we have only to compute $\sigma +^M \tau$, which can be done in time $k(|\sigma| + |\tau|)^4$ by Lemma 4.9. Now, since, for any integers x and y greater than 1, $x + y \leqslant xy$, it follows that for some c,

$$T_P(E\sigma, E\tau) \leqslant c\,|E\sigma|^4\,|E\tau|^4.$$

(2) To compute $AE\sigma\tau \cdot^M \rho$, we need to multiply both $E\sigma$ and τ by ρ and then add the results ρ_1 and ρ_2. By Lemma 4.9, $|\rho_1| + |\rho_2| \leqslant |E\sigma|\,|\rho| + |\tau|\,|\rho| < |AE\sigma\tau|\,|\rho|$. It now follows from Lemma 4.9 that, for some k,

$$T_P(AE\sigma\tau, \rho) \leqslant T_P(E\sigma, \rho) + T_P(\tau, \rho) + k\,|AE\sigma\tau|^4\,|\rho|^4.$$

Now by induction, we know that $T_P(E\sigma, \rho) \leq c\,|E\sigma|^5\,|\rho|^5$ and that $T_P(\tau, \rho) \leq c\,|\tau|^5\,|\rho|^5$. Then taking $c \geq k$, we have

$$\begin{aligned} T_P(AE\sigma\tau, \rho) &\leq c\,|E\sigma|^5\,|\rho|^5 + c\,|\tau|^5\,|\rho|^5 + c\,|AE\sigma\tau|^4\,|\rho|^4 \\ &\leq c\,|\rho|^5\,[(|E\sigma| + |\tau|)^5 + |AE\sigma\tau|^4] \\ &\leq c\,|\rho|^5\,|AE\sigma\tau|^5. \qquad \text{(by Lemma 4.8).} \end{aligned}$$

This completes the proof of Lemma 4.10. □

Let us note that of course, for any σ, $E^{\mathcal{M}}(\sigma) = E\sigma$, so that the function $E^{\mathcal{M}}$ is linear-time. It is also true that the iterations $EE, EEE, \ldots$ of E, which are the interpretations of the functions $2^x, 2^{2^x}, \ldots$ are also linear-time.

This completes the proof of Theorem 4.1. □

Let us note that the upper bounds given for the computation time of the various functions are not necessarily the best possible.

Now we may view the model $\mathcal{M}$ as a new representation of the natural numbers in which the computation of exponentiation is very easy. Of course, the computations of successor, addition and multiplication are more difficult than in the standard unary or binary representation. A natural question is the efficiency of the representation $\sigma(n)$ of the number n.

Let us say that $\mathcal{N}$ is the standard binary representation of the natural numbers. It is easy to see that the function $n(\sigma)$ from $\mathcal{M}$ into $\mathcal{N}$ cannot be polynomial time, because it maps the short strings E^k0 onto the long strings $2^{2^{2^{\cdots^k}}}$. On the other hand, it is not hard to show that the function $\sigma(n)$ from $\mathcal{N}$ into $\mathcal{M}$ is polynomial time. We now show that the representation $\sigma(n)$ of the natural numbers n is not too much longer than the binary representation. Let $|n|$ be the length of the binary representation of n (that is, roughly, $\log_2 n$).

Proposition 4.11. *For any natural number n, $|\sigma(n)| \leq 2\,|n|^2$.*

Proof. Note first that for $k > 0$, $|k| \leq k$ and that for $k < 4$, $|k|^2 \leq 2k$. The proof is by induction on n. There are three cases.

(0) For $n \leq 32$, this may be checked by a table.

(1) For $n = 2^k > 1$, we have $\sigma(n) = E\sigma(k)$ and $|n| = k + 1$, so that

$$\begin{aligned} |\sigma(n)| &= 1 + |\sigma(k)| \\ &\leq 1 + 2\,|k|^2 \qquad \text{(by induction)} \\ &\leq 1 + 2k^2 \leq 2(k+1)^2 = 2\,|n|^2. \end{aligned}$$

(2) For $n = 2^k + m$ with $0 < m < 2^k$ and $k > 4$, we have $\sigma(n) = AE\sigma(k)\sigma(m)$,

$|m| \leq k$ and $|n| = k + 1$, so that

$$\begin{aligned} |\sigma(n)| &= 2 + |\sigma(k)| + |\sigma(m)| \\ &\leq 2 + 2|k|^2 + 2|m|^2 \qquad \text{(by induction)} \\ &\leq 2 + 4k + 2k^2 \qquad \text{(by the note above)} \\ &= 2(k+1)^2 = 2|n|^2. \qquad \square \end{aligned}$$

It appears that the function 3^x is not polynomial-time in our model, because the length of $\sigma(3^x)$ is in general too long. However, it seems reasonable that the idea of Theorem 4.1 can be modified to give a model where 3^x is p-time, although perhaps 2^x is no longer p-time.

Notice that in the structure, an iterated power n of two has a very short representation, whereas $n - 1$ may have a long representation. For example, the number sixteen is represented by $EEEEE0$, whereas fifteen is represented by $AEEEE0EEE0EE0E0$. Thus it is not hard to see that the subtraction function is not polynomial-time in the structure. It is possible to modify the structure to include subtraction while remaining polynomial-time.

Define the subtraction function $m - n$ on $\mathbb{N}$ as usual, so that $m - n = r$ if $n + r = m$ or if $m < n$ and $r = 0$. Then we have the following.

Theorem 4.12. *The structure* $(\mathbb{N}, S, +, -, \cdot, 2^x, <, 0)$ *for the language consisting of three unary function symbols, two binary function symbols, one binary relation symbol and one constant symbol, is recursively isomorphic to a polynomial-time structure.*

Proof. The proof is similar to that of Theorem 4.1. We will define the structure and make a few remarks about the details of the proof. We begin by adding a symbol I to the language of arithmetic to represent subtraction. Now we let

$$n(I\sigma\tau) = n(\sigma) - n(\tau).$$

The representation $\sigma(n)$ is modified so that

$$\sigma(2^k + m) = AE\sigma(k)\sigma(m), \quad \text{for } 0 < 2m \leq 2^k$$

and

$$\sigma(2^{k+1} - m) = IE\sigma(k+1)\sigma(m), \quad \text{for } 0 < 2m < 2^k.$$

The modified set M^- of terms will again be the set of images $\sigma(n)$ of natural numbers and the modified structure will be defined once again to make σ an isomorphism of the structures.

It then remains to prove that the ordering relation and all of the operations are polynomial-time.

The main difficulty arises in the proof of Lemma 4.6. We first have to prove that the relations "$n(\sigma) = n(\tau) + 1$" and "$n(\sigma) = 2 \cdot n(\tau)$" are both polynomial-time. This is because, for example, $AE\sigma\tau \in M^-$ if and only if $2 \cdot n(\tau) \leq n(AE\sigma)$.

The remaining lemmas go through as before (only with twice as many cases). □

References

[1] H. Enderton, A Mathematical Introduction to Logic (Academic Press, New York, 1986).
[2] S.S. Goncharov, Some properties of the constructivization of Boolean algebras, Sib. Mat. Zh. 16 (1975) 203–214.
[3] S. Grigorieff, Every recursive linear ordering has a copy in DTIME(n), J. Symbolic Logic 55 (1990) 260–276.
[4] J.E. Hopcroft and J.D. Ullman, Formal Languages and Their Relations to Automata (Addison-Wesley, Reading, MA, 1969).
[5] A. Nerode and J. Remmel, Complexity theoretic algebra II: the free Boolean algebra, Ann. Pure Appl. Logic 41 (1989) 71–99.
[6] J. Remmel, Recursive Boolean algebras, in J.D. Monk, ed., Handbook of Boolean Algebras (North-Holland, Amsterdam, 1989) 1099–1165.